НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

«На правах рукопису»

УДК 04.004

До захисту допущено:
Завідувач кафедри
_____ О.Л. ТИМОЩУК
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Системний аналіз фінансового ринку»

спеціальності 124 «Системний аналіз»

на тему: « Методи та засоби прогнозування індексу акцій у реальному часі на базі хмарних обчислювальних сервісів»

Виконав:
студент II курсу, групи КА-02мп
Величко Георгій Вячеславович _____

Керівник:
в.о. завідувача кафедри системного проектування
КПІ ім. Ігоря Сікорського, д.т.н., проф. Мухін В.Є.

_____

Рецензент:
професор кафедри інформаційних систем та технологій
КПІ ім. Ігоря Сікорського
д.т.н., доц., Я.І. Корнага

_____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ
2021

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

Рівень вищої освіти – другий (магістерський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз фінансового ринку»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ О.Л. ТИМОЩУК

«___»_____20__ р.

ЗАВДАННЯ

на магістерську дисертацію студенту

Величку Георгію Вячеславовичу

1. Тема дисертації: «Методи та засоби прогнозування індексу акцій у реальному часі на базі хмарних обчислювальних сервісів», науковий керівник дисертації Мухін Вадим Євгенович, д.т.н., проф., в.о. завідувача кафедри СП затверджені наказом по університету від «_02_»_листопада_ 20_21_ р. №_3651-с__

2. Термін подання студентом роботи: 12 грудня 2021 року
3. Об'єкт дослідження: щохвилинні дані про індекс S&P 500 за календарний рік.
4. Предмет дослідження: багатошарова нейронна мережа, згорткова нейронна мережа та мережа довгої короткочасної пам'яті, хмарні обчислювальні сервіси, Amazon Web Services.
5. Перелік завдань, які потрібно розробити:
    1) Здійснити огляд технічної літератури за темою роботи;
    2) Дослідити актуальність обраної теми;
    3) Ознайомитись із існуючими методами прогнозування індексу акцій.
    4) Ознайомитись із існуючими рішеннями хмарних обчислювальних сервісів.
    5) Дослідити можливість реалізації системи прогнозування в реальному часі на базі хмарних сервісів.
    6) Розробити модель системи прогнозування в реальному часі на базі хмарних сервісів та описати її компоненти.
    7) Побудувати нейронні мережі та спрогнозувати тренд індексу акцій;

8) Провести аналіз результатів.
9) Провести аналіз ринкових можливотсей запуску стартап проекту;
10) Розробити концептуальні висновки;

6. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо):

7. Дата видачі завдання: 5 вересня 2021

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання магістерської дисертації	Примітка
1	Отримання завдання	01.09.2021 – 05.09.2021	Виконано
2	Збір інформації	06.09.2021 – 12.09.2021	Виконано
3	Ознайомлення з літературою і підготовка теоретичної частини магістерської дисертації	13.09.2021 – 26.09.2021	Виконано
4	Аналіз вимог завдання, вибір методів і засобів розв’язання поставленої задачі	27.09.2021 – 3.10.2021	Виконано
5	Розробка програмного продукту	4.10.2021 – 31.10.2021	Виконано
6	Проведення аналізу ринкових можливостей стартап-проекту	1.11.2021 – 7.11.2021	Виконано
7	Отримання висновків після тестування	8.11.2021 – 15.11.2021	Виконано
8	Оформлення магістерської дисертації	15.11.2021 – 12.12.2021	Виконано
9	Отримання допуску до захисту та подача роботи до ДЕК	13.12.2021 – 14.12.2021	Виконано

Студент Георгій ВЕЛИЧКО

Керівник Вадим МУХІН

РЕФЕРАТ

Магістерська дисертація: 92 с., 28 рис., 21 табл., 1 дод., 30 джерел.

ІНДЕКС АКЦІЙ, БАГАТОШАРОВІ НЕЙРОННІ МЕРЕЖІ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, НЕЙРОННІ МЕРЕЖІ ДОВГОЇ КОРОТКОЧАСНОЇ ПАМ’ЯТІ, PYTHON, ХМАРНІ ОБЧИСЛЮВАЛЬНІ СЕРВІСИ, AMAZON WEB SERVICES, GOOGLE CLOUD PLATFORM

Тема роботи — Методи та засоби прогнозування індексу акцій у реальному часі на базі хмарних обчислювальних сервісів.

Об’єкт дослідження — методи прогнозування акцій на базі хмарних обчислювальних сервісів.

Предмет досліджень — багатошарова нейронна мережа, згорткова нейронна мережа та мережа довгої короткочасної пам’яті, хмарні обчислювальні сервіси, Amazon Web Services.

Мета роботи — підвищити ефективність прогнозування індексу акцій на базі хмарних обчислювальних сервісів та розробити модель реалізації системи на базі хмарних сервісів.

Актуальність — використання апарату штучних нейронних мереж дає змогу робити більш точні передбачення ціни акцій порівнюючи з іншими технічними методами. Хмарні сервіси забезпечують швидке та гнучке розгортання інфраструктури, забезпечує їх повсякчасну доступність.

В ході виконання роботи було реалізовано декілька ахітектур нейронних мереж та проведено аналіз результатів їх роботи. Також було спроектовано систему прогнозування індексу акцій в реальному часі на базі хмарних сервісів.

Для покращення результатів у майбутньому можна застосувати Keras Tuner, який автоматично підбирає гіперпараметри нейронної мережі. Також можна використовувати додаткову інформацію про компанію і додавати ці дані до моделей.

ABSTRACT

Master`s thesis contains: 92 p., 21 tables, 28 fig., 1 add. and 30 references.

STOCK INDEX, MULTILAYER NEURAL NETWORK, CONVOLUTIONAL NEURAL NETWORK, LONG SHORT-TERM MEMORY, PYTHON, CLOUD COMPUTE SERVICES, AMAZON WEB SERVICES, GOOGLE CLOUD PLATFORM

The theme: The mechanisms tools for real-time stock market index prediction based on Cloud Compute Services.

The objects of this research are mechanisms for stock market index prediction based on Cloud Compute Services.

The subjects of this research are neural network architectures: multilayer neural network, convolutional neural network, long short-term memory, cloud compute services, Amazon Web Services.

The purpose of this work is to improve efficiency of stock market index prediction based on Cloud Compute Services and design model of system for stock market index forecasting based on Cloud Compute Services.

The relevance of this topic is that using neural networks could drastically improve stock market values forecasting comparing to other technical methods. Cloud compute services provides fast and flexible deployment of infrastructure as well as constant availability. Also designed a model of system for real-time stock index forecasting based on cloud compute services.

During research I built three architectures of neural networks and analyzed which of them work better with the given case.

For further research, it is possible to use Keras Tuner which allows to automatically tune hyperparameters on our neural network and improve output result. Also, additional data about companies provided real-time can improve models.

ЗМІСТ

ВСТУП

Як відомо, купівля, продаж та емісія акцій здійснюється інвесторами на фондовому ринку. Оскільки це ринок — то ціна на даний вид цінних паперів не є фіксованою і може залежати від величезної кількості факторів. Звісно, учасники торгів на фондовому ринку мають бажання отримати значний прибуток від підвищення або зниження цін на акції певних компаній: якщо інвестор знає, що ціни акцій будуть підвищуватись — він може купити ці акції, або ж дати їх на короткий строк у борг; якщо інвестор знає, що ціни на дані акції будуть знижуватись — він може їх продати, або ж взяти у борг, продати по високій ціні і повернути борг, коли ціни на дані акції впадуть. Проте, як відомо — будь-яка інвестиційна діяльність пов’язана з певними ризиками. Звісно, у цій галузі частина успіху залежить від удачі інвестора. Проте, більшість досвідчених та успішних інвесторів не покладатись на цю складову. Для досягнення успіху і отримання прибутку необхідно провести ґрунтовний аналіз та зробити якомога точний прогноз.

Існує декілька підходів для здійснення аналізу фондового ринку. Одні з них базуються на історичних даних про ціни акцій. Інші базуються на аналізі даних про навколишній світ та події у ньому, фінансовій звітності певних компаній, макроекономічних показниках та інших доступних з відкритих джерел даних. Використання апарата штучних нейронних мереж дає змогу об’єднати ці підходи в один. Вони можуть приймати на вхід різну за природою інформацію. Це дає змогу знаходити неочевидні закономірності й саме тому інвестори часто використовують цей метод.

Використання апарату штучних нейронних мереж дає змогу робити більш точні передбачення ціни акцій порівнюючи з іншими технічними методами. Хмарні сервіси забезпечують швидке та гнучке розгортання інфраструктури, забезпечує їх повсякчасну доступність.

Об’єктом дослідження слугують методи прогнозування акцій на базі

хмарних обчислювальних сервісів. Предметом дослідження є багатошарова нейронна мережа, згорткова нейронна мережа та мережа довгої короткочасної пам’яті, а також хмарні обчислювальні сервіси, зокрема Amazon Web Services.

Мета роботи — підвищити ефективність прогнозування індексу акцій на базі хмарних обчислювальних сервісів та розробити модель реалізації системи на базі хмарних сервісів.

Наукова новизна полягає в розробці методу прогнозування індексу акцій на базі хмарних обчислювальних сервісів.

РОЗДІЛ 1 ОГЛЯД ЗАСОБІВ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ІНДЕКСУ АКЦІЙ ТА ХАРНИХ РЫШЕНЬ ДЛЯ БАЗУВАННЯ СИСТЕМИ

1.1 Огляд предметної області

У процесі росту компанії, керівництво може перевести компанію з приватної форми власності у публічну. Це робиться з метою формування статутного капіталу. Для цього компанія має здійснити емісію акцій на одну із фондових бірж (зазвичай акції однієї компанії існують лише в межах однієї фондової біржі). У майбутньому компанія може робити емісію акцій з метою залучення додаткових інвестицій для розвитку компанії.

Акція — цінний папір без визначеного часу обігу, що засвідчує участь його власника у статутному капіталі акціонерного товариства, дає власникові право на одержання частини прибутку у вигляді дивіденду, а також на участь у розподілі майна в разі ліквідації товариства [1]. Можна виділити три основні операції над акціями: купівля, продаж та емісія. Усі операції над акціями здійснюються на фондовому ринку.

Фондовий ринок — сукупність учасників фондового ринку та правовідносин між ними щодо розміщення, обігу та обліку цінних паперів і похідних (деривативів) [2]. Можна виділити два основних учасники ринку: емітенти (ті, хто здійснюють випуск цінних паперів) та інвестори (учасники, що здійснюють купівлю і продаж цінних паперів).

Головне питання: навіщо людям купувати цінні папери на фондових ринках? Можна виділити три основні напрями.

Перший напрям — класичні інвестиції. Традиційні інвестори здійснюють купівлю акцій з упевненістю, що вони зростуть у ціні в майбутньому. Цей напрям діяльності передбачає довгострокові інвестиції. Інвестор може отримати дохід із продажу акцій по вищій ціні, або ж з періодичного отримання виплат за цінними паперами [3].

Другий напрям — спекуляція, або ж трейдинг. На сьогоднішній день більшу частину учасників ринку складають трейдери. Цей вид інвесторів

заробляє на коливанні ціни цінних паперів. Зазвичай це короткострокові інвестиції. Трейдер діє за стратегією, яка передбачає швидке придбання акцій по низькій ціні й перепродаж їх по вищій, або ж, якщо трейдер знає, що ціна на акцію буде падати — він може взяти акції у борг коли вони мають високу ціну, продати їх і повернути коли ціна на дані акції впаде. Цей напрям інвестиційної діяльності може принести значно більші прибутки, аніж класичний напрям інвестицій, але він пов'язаний зі значно більшими ризиками [4].

Третій напрям — це ведення бізнесу на просторах майданчику фондового ринку. За допомогою таких інструментів, як опціони і ф'ючерси, бізнесмени можуть застрахувати себе від коливання ціни на товари. Це привносить деяку стабільність у процес ведення бізнесу [5].

1.2 Аналіз сучасних засобів прогнозування індексу акцій

Аналіз і прогнозування є обов'язковими складовими діяльності інвестора при роботі з фондовим ринком. Наявність сукупності різних видів прогнозів дозволяє значно зменшити ризики невдалих інвестицій які можуть призвести до великих збитків. Раніше при вкладанні грошей інвестори не мали змоги обробити величезний обсяг інформації і в багатьох випадках покладались на обмежену кількість критеріїв та показників, а іноді навіть на якісь свої власні суб'єктивні відчуття. Іноді це давало результат [6]. Проте на сьогоднішній день, із розвитком сучасних технологій обробки і зберігання інформації, які є доступні буквально кожній людині, а також шаленій конкуренції між різними інвесторами, такий підхід не є робочим і призводить до радше збитків, аніж постачання якого-неякого стабільного доходу. Проведення даного аналізу інвесторами необхідно для максимізації прибутку від вкладених грошей в процесі коливання цін на певні види цінних паперів. Будь яке прогнозування базується на принципі що ті тенденції, ті фактори, що на них впливали і ті залежності між ними у минулому

та теперішньому часі збережуться і у майбутньому. І звісно, прогнозування індексу акцій не є виключенням з цього принципу [7].

Як і у будь-якому іншому прогнозі, прогноз індексу акцій є не повністю достовірним і має радше ймовірнісний характер. Достовірність цього прогнозу у більшості залежить від набору факторів які враховує інвестор. Наприклад, невраховування потенційно важливих факторів може істотно зменшити точність прогнозу, але якщо враховувати недоцільні фактори, то можна теж зменшити точність прогнозу шляхом знаходження безглуздих кореляцій. Не варто забувати і про швидку мінливість фондового ринку, необхідність постійно корегувати свій прогноз зважаючи на надходження актуальних даних та враховувати динаміку окремих його складових.

Можна виділити декілька основних методів для побудови прогнозу: фундаментальний аналіз, технічний аналіз, метод експертних оцінок, а також статистичні методи [8].

1.3 Загальний огляд основних провайдерів хмарних обчислювальних сервісів

Модель хмарного обчислення являє собою модель, у якій за вимогою користувача йому надається швидкий і зручний мережевий доступ до пулу обчислювальних ресурсів. Користувачу цих сервісів надаються широкі можливості з налаштування об’єктів обчислювальної інфраструктури, таких як сервери, мережеві сервіси та сховища даних, а також можливості миттєвого масштабування цієї інфраструктури. Перевагою таких сервісів є відсутність необхідності в утримуванні, налаштуванні та оновленні інфраструктури, окрім того відсутність необхідності платити за інфраструктуру, яку користувач не використовує і можливість практично необмежено збільшувати розміри цієї інфраструктури.

В останні роки, модель хмарних обчислювальних набула неабиякої популярності серед багатьох провідних копаній-розробників програмного забезпечення, як приватних, так і державних. Тож не дивно, що високий попит на цьому ринку породив і багато пропозицій. За даними веб порталу з обробки даних statista.com [9], на перший квартал 2021 року можна виділити три основних лідера у цьому сегменті ринку цифрових послуг (див. рис. 1.1).:

- Amazon Web Services (AWS).
- Microsoft Azure (Azure).
- Google Cloud Platform (GCP).

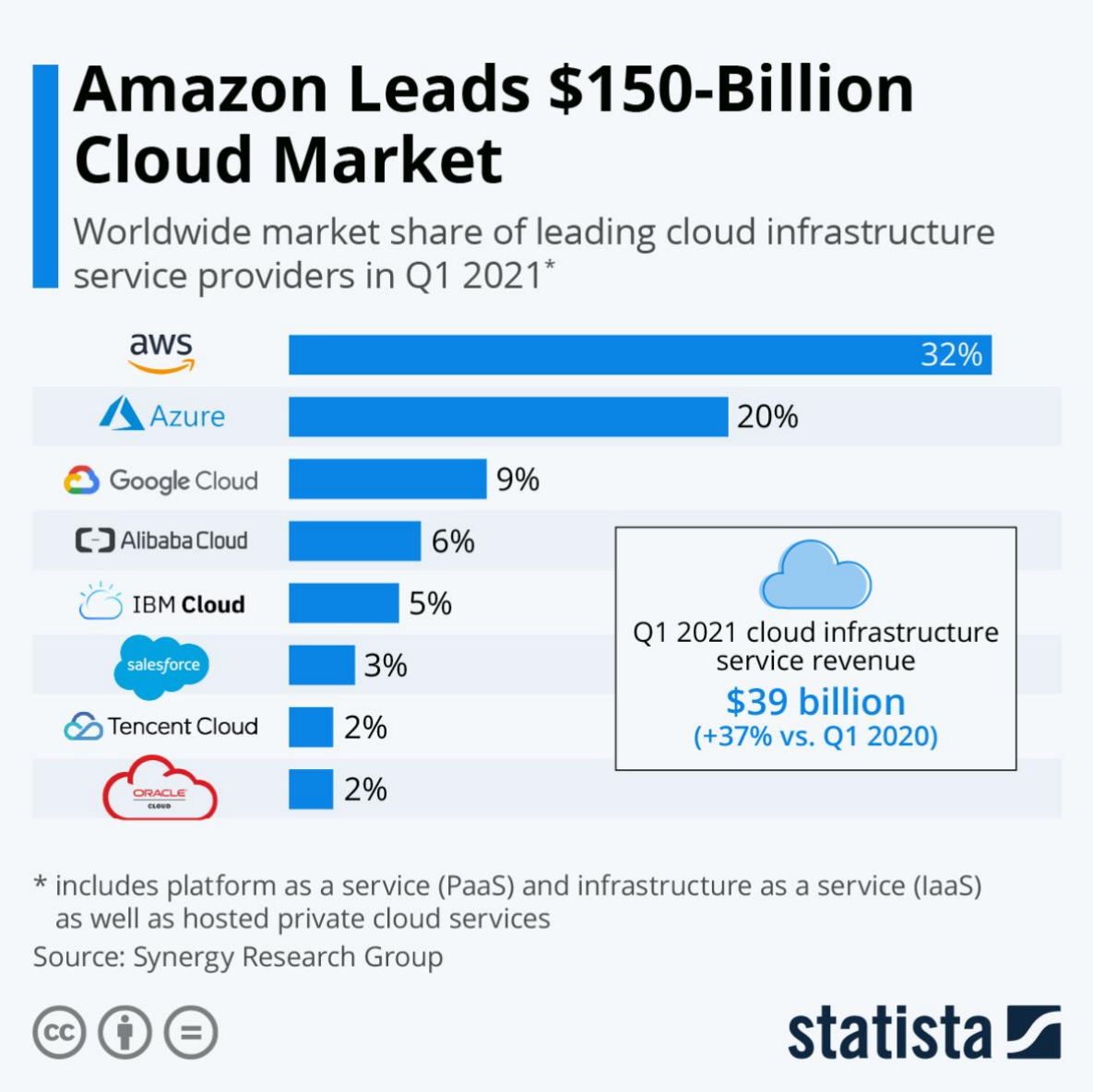

Рисунок 1.1 - Розподіл ринку хмарних обчислювальних сервісів станом на перший квартал 2021 року за даними statista.com [9]

Загальний об’єм даного ринку, станом на перший квартал 2021 року, склав приблизно 150 мільярдів доларів США. Як можна побачити з Рисунка 1.1, лідером ринку є Amazon Web Services, частка якого складає 32%. Цей сегмент ринку, а саме хмарні обчислювальні сервіси, є одними з найбільших сегментів бізнесу компаній, які ними володіють. Попри те варто зазначити, що для жодного з лідерів ринку цей сегмент не є основним сегментом їх діяльності. У наступних підрозділах розглянемо детальніше кожного з найбільших гравців цього ринку.

1.3.1 Amazon Web Services

Amazon Web Services (поширена скорочена назва - AWS) є одною з найпопулярніших платформ для розробки програмного забезпечення. Цей сервіс є дочірньою компанією мега корпорації Amazon.com Inc., яка за даними сайту statista.com [10] посідає 4 місце у списку найдорожчих корпорацій світу. До речі, інші дві компанії, які володіють найбільшими хмарними обчислювальними сервісами, а саме Alphabet Inc. Та Microsoft Inc., які володіють Google Cloud Platform (GCP) та Microsoft Azure відповідно, разом із Amazon.com Inc. Входять до п’ятірки найдорожчих компаній світу за даними того ж веб ресурсу statista.com (див. рис. 1.2).

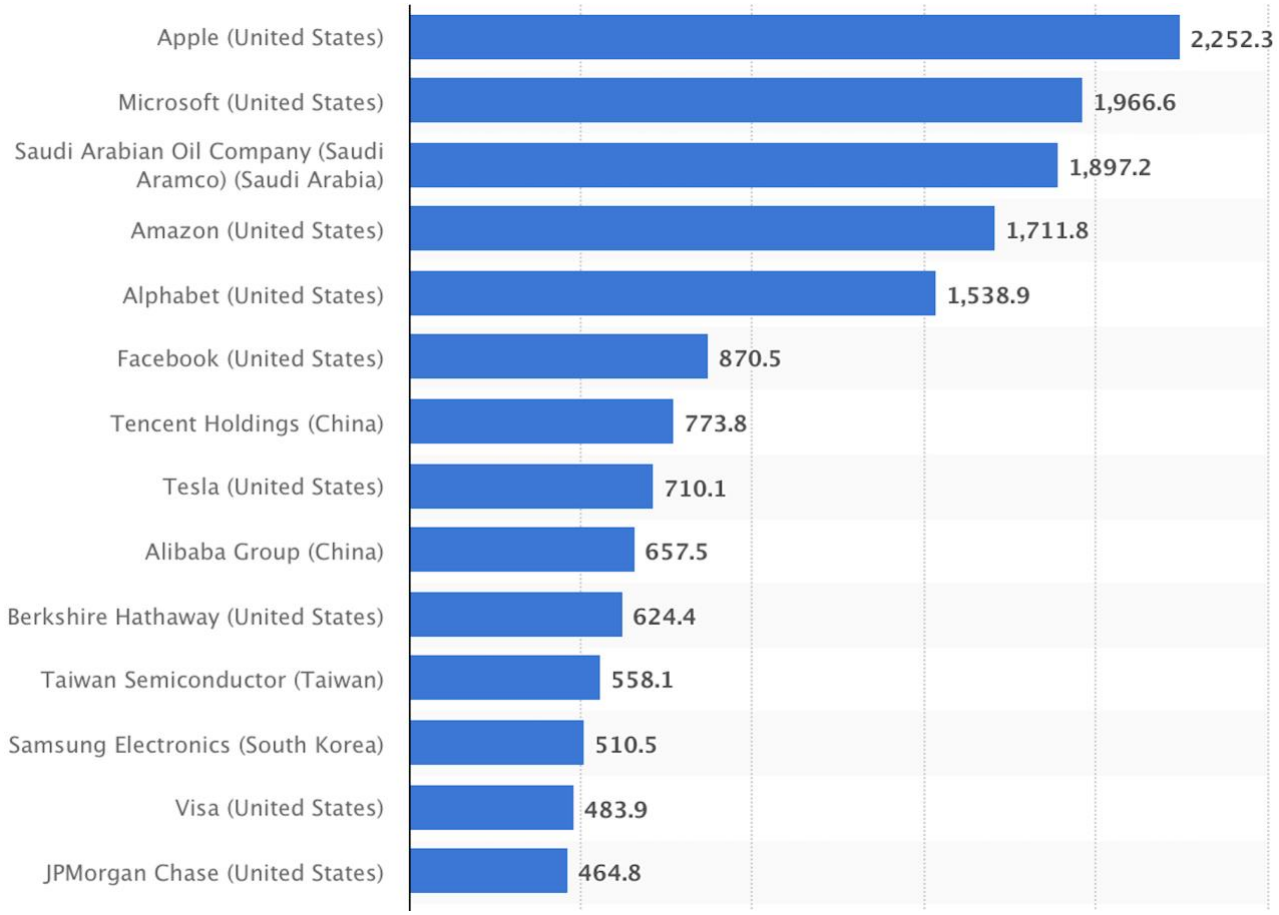

Рисунок 1.2 - Список найдорожчих компаній світу за даними веб ресурсу обробки даних statista.com [10]

Amazon Web Services надає послуги оренди платформи для хмарних обчислень фізичним особам, приватним компаніям та урядовим установам за моделлю платної підписки.

Фізично Amazon Web Services втілений у величезних дата-центрах, які знаходяться у 25 локаціях (у термінології власника мають назву “регіони”) розташованих по всьому світу (див. рис. 1.3).

Рисунок 1.3 - Основні локації розташування дата центрів Amazon Web Services нанесені на мапу світу. [11]

Децентралізація обчислювальних потужностей дозволяє знизити ризики повної відмови системи, а також надає можливість отримувати відповідь на запит зі значно нижчою затримкою. Слід також зазначити, що обсяг наявних послуг відрізняється в залежності від обраного регіону, а також ціна на одні й ті самі послуги не є однаковою в залежності від обраного регіону. Проте доступ до обчислювальних потужностей дата центру у будь-якому регіоні можливий практично з будь-якого куточку світу, тож його вибір в більшості випадків не залежить від розташування розробника. У таблиці 1.1 наведено порівняння цін на аналогічні послуги у різних регіонах AWS.

Таблиця 1.1 - Порівняльна таблиця цін на основні ресурси, які надаються Amazon Web Services у більшості регіонів світу, у USD$ [12]

Updated : 22.10.21	Description		Standard, 100TB	4 vCPUs 50 MB 30GB SSD	MySQL 30GB db.m5.12xlarge	500MB 1 bil requests 1s	
	Region Code	Region	AWS S3	EC2 (Linux)	RDS	Lambda	Total
	us-east-2	Ohio	2,304.00		5,998.74	8,331.17	8,302.74
	us-east-1	North Virginia	2,304.00	153.23	5,998.74	8,331.17	8,455.97
	us-west-2	Oregon	2,304.00	153.23	5,998.74	8,331.17	8,455.97
	eu-north-1	Stockholm	2,304.00	199.35	6,298.34	8,331.17	8,801.69
	eu-west-1	Ireland	2,304.00	183.24	6,630.15	8,331.17	9,117.39
	ca-central-1	Canada	2,508.80	206.39	6,630.15	8,331.17	9,345.34
	eu-west-2	London	2,406.40	221.31	6,945.90	8,331.17	9,573.61
	eu-south-1	Milan	2,406.40	248.01	6,961.96	9,751.85	9,616.37
	us-west-1	N. California	2,611.20	209.61	6,841.08	8,331.17	9,661.89
	eu-central-1	Frankfurt	2,457.60	227.24	7,121.31	8,331.17	9,806.15
	me-south-1	Bahrain	2,508.80	289.06	7,293.02	10,332.65	10,090.88
	af-south-1	Cape Town	2,744.32	312.72	7,890.11	11,051.91	10,947.15
	ap-southeast-2	Sydney	2,508.80	225.81	8,242.68	8,331.17	10,977.29
	ap-south-1	Mumbai	2,508.80	227.27	8,452.50	8,331.17	11,188.57
	ap-northeast-3	Osaka	2,508.80	227.24	8,705.50		11,441.54
	ap-east-1	Hong Kong	2,508.80	249.24	9,066.84	11,461.96	11,824.88
	eu-west-3	Paris	2,406.40	248.03	6,945.90	8,331.17	17,931.50
	ap-northeast-1	Tokyo	2,508.80	227.27	8,242.68	8,331.17	19,309.92
	ap-northeast-2	Seoul	2,508.80	227.24	8,277.72	8,331.17	19,344.93
	sa-east-1	Sao Paulo	4,070.40	301.5	8,107.35	8,331.17	20,810.42

Дата-центр – це будівля або приміщення, призначенням яких є зберігання та обробка даних. У межах одного регіону існує декілька менших дата-центрів, які отримали назву зони доступності (availability zone). Між цими зонами доступності існує високошвидкісне з'єднання, тож компанія, яка надає послуги хмарного обчислення, гарантує наднизьку затримку у доставці сигналів між цими зонами (див. рис. 1.4). Увесь трафік між цими зонами доступності зашифрований, а відстань між зонами доступності у межах одного регіону не перевищує 100 кілометрів. Кожна зона доступності має автономне джерело живлення та охолодження, а також ретельно охороняється.

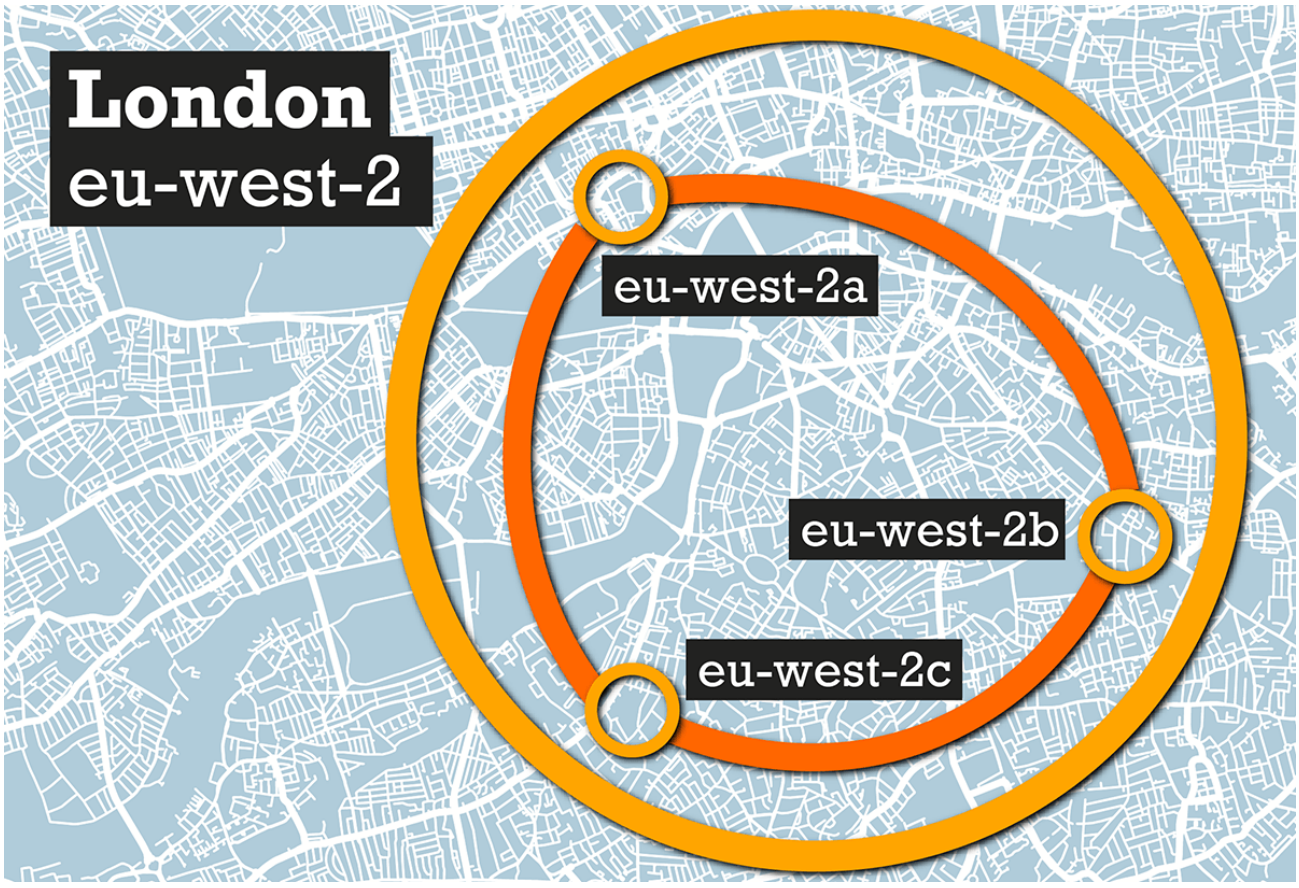

Рисунок 1.4 - Схематичне зображення зон доступності Amazon Web Services у регіоні eu-west-2, який розташований у Лондоні. [13]

Дата центр Amazon Web Services умовно поділений на 4 рівня:

- Рівень фізичного периметру. Фізична безпека центру обробки даних AWS починається на рівні периметра. Цей рівень включає ряд функцій безпеки

залежно від місця розташування, таких як охоронці, огорожі, канали безпеки, технологія виявлення вторгнень та інші заходи безпеки.

- Рівень інфраструктури. Інфраструктурний рівень - це будівля центру обробки даних, обладнання та системи, які підтримують його роботу. Такі складові, як резервне енергетичне обладнання, система ОВКП (Опалення, вентиляція та кондиціювання повітря) та обладнання для пожежогасіння, є частиною інфраструктурного рівня. Ці пристрої та системи допомагають захистити сервери і, зрештою, дані користувачів.
- Рівень даних. Рівень даних є найважливішою точкою захисту, оскільки це єдина область, де зберігаються дані клієнтів. Захист починається з обмеження доступу та збереження розподілу привілеїв для кожного рівня. Крім того, існують пристрої для виявлення загроз, відеоспостереження та системні протоколи, які додатково захищають цей рівень.
- Рівень екології. Екологічний рівень присвячений екологічним міркуванням від вибору ділянки та будівництва до експлуатації та стабільності. AWS ретельно вибирає місця розташування центрів обробки даних для зменшення екологічних ризиків, таких як повені, екстремальні погодні умови та сейсмічна активність.

На рисунку 1.5 можна бачити, як усі 4 рівнів поєднуються в одному дата центрі.

Рисунок 1.5 - Схематичне зображення типового центру обробки даних Amazon Web Services [13]

Доступ до кожного з рівнів є обмеженим і базується на потребах бізнесу. Завдяки імплементації моделі порівневої перевірки рівня доступу, доступ до кожного наступного рівня не надається за замовчуванням. Доступ до будь-якого конкретного рівня надається лише за наявності певної потреби в доступі до цього конкретного рівня. Водопостачання, електропостачання, телекомунікації та підключення до Інтернету спроектовані із запасом, щоб мати змогу підтримувати безперервну роботу в умовах надзвичайної ситуації. Електричні системи живлення розроблені таким чином, щоб вони були повністю резервними, щоб у разі перебоїв в роботі могли бути задіяні блоки безперебійного живлення для певних функцій, тоді як генератори можуть забезпечувати резервне живлення для всього об’єкта. Люди та системи контролю перевіряють та контролюють різні параметри, як температуру та вологість, щоб запобігти перегріванню, що ще більше зменшує можливі перебої в обслуговуванні.

Сервери Amazon Web Services являють собою одну велику систему, тож коли користувач робить запит на виділення йому певних ресурсів ці ресурси виділяються на логічному рівні, а не на фізичному.

До десятки найпопулярніших сервісів Amazon Web Services входять наступні сервіси [14]:

1. Amazon S3 (Simple storage service). Даний сервіс призначений для зберігання та завантаження даних із хмари. S3 дозволяє користувачеві зберігати, завантажувати та отримувати великі файли розміром до 5 ТБ з хмари. Це масштабований, недорогий і швидкісний сервіс, призначений для архівування та резервного копіювання прикладних програм та даних. Користувачі мають контроль над публічним або приватним доступом до даних.
2. Amazon EC2 (Elastic Computing Cloud). Даний сервіс надає обчислювальні можливості в хмарі Amazon Web Services з можливістю масштабування. За допомогою Amazon EC2 можна швидко, ефективно та не дорого розробляти

та розгортати програмні продукти. Також можна використовувати Amazon EC2 для запуску віртуальних серверів відповідно до вимог користувача.

3. Amazon Lambda. Даний сервіс дозволяє користувачеві запускати код без будь-якого сервера, тобто втілює без серверну концепцію (Serverless). Amazon Lambda виконує код лише тоді, коли це потрібно користувачеві та автоматично масштабує його. Користувачі платять лише за час обчислень, не потрібно платити у той час, коли його код не працює. Ця служба підтримує код, написаний на Node.js, Java, Python та мовах, які підтримуються Amazon Linux.
4. Amazon Glacier. Це он-лайн сервіс для зберігання даних, який надає недороге та ефективне сховище з функціями безпеки для архівування та резервного копіювання даних. За допомогою Glacier ви можете ефективно зберігати інформацію протягом місяців, років або навіть десятиліть. За перевагу у вигляді низької ціни користувач розплачується часом, за який він може отримати доступ до цих даних. Amazon Web Services гарантує відповідь на запит про доступ до даних в межах одного тижня (у середньому три дні).
5. Amazon SNS (Simple Notification Service). SNS розшифровується як служба простого сповіщення. Даний сервіс керує та доставляє повідомлення або сповіщення користувачам та клієнтам. У SNS існує два типи клієнтів: абоненти та видавці. Видавці створюють та надсилають повідомлення до абонента по каналах зв’язку. Абоненти отримують сповіщення від видавця щодо одного з підтримуваних протоколів, таких як Amazon SQS, HTTP/S, EMAIL, SMS, Amazon Lambda, тощо.
6. Amazon CloudFront. Це варіація концепції CDN (Content Delivery Network), яка полягає в тому, що файли попередньо завантажені на сервери у різних точках світу, тож користувач може значно швидше завантажити ці файли. Цей сервіс прискорює надсилання вашим користувачам вашого динамічного та статичного контенту, такого як .css, .html та файли зображень.
7. Amazon EBS (Elastic Block Storage). Використовується для зберігання постійних даних. Це сховище на рівні блоків для використання в

обчислювальних серверів EC2. Ви можете використовувати службу EBS для переміщення даних з одного сервера в інший, не втрачаючи збережених даних. Ви можете монтувати кілька сховищ до одного сервера, але кожен сховище можна приєднати лише до одного сервера одночасно.

8. Amazon Kinesis. AWS пропонує сервіс Amazon Kinesis для обробки великих даних у режимі реального часу. Це дозволяє розробникам брати будь-який великий обсяг даних з будь-якого джерела, яке може працювати на серверах EC2. Він зберігає, збирає та обробляє дані з великих розподілених потоків, таких як канали соціальних мереж та події журналу. Після завершення обробки даних він одночасно розповсюджує дані споживачам.
9. Amazon VPC (Virtual Privat Cloud). Даний сервіс дозволяє ізолювати частину хмарних сервісів користувача. Цей сервіс дає повний контроль над віртуальним мережевим середовищем, а також дає можливість створювати та розподіляти ресурси всередині хмари користувача. Також він відповідає за безпеку ресурсів користувача.
10. Amazon SQS (Simple Queue Service). Розшифровується як сервіс простих черг і управляє сервісами черги повідомлень. За допомогою цього сервісу користувач може переміщувати дані або повідомлення з однієї програми в іншу, навіть якщо вони не перебувають у запущеному чи активному стані. SQS надсилає повідомлення між кількома службами, включаючи S3, DynamoDB, EC2 сервер, а також використовує службу черги повідомлень Java для доставки інформації. Максимальний час перебування видимості повідомлення становить 12 годин у черзі SQS.

Детальніше про сервіси у наступних підрозділах.

1.3.2 Google Cloud Platform

Компанія Google (нині Alphabet Inc.) займає домінуюче положення у багатьох галузях пов'язаних з всесвітньою мережею:

- Найбільший пошуковий сервіс - Google Search.
- Найбільша платформа для поширення та перегляду відео – YouTube.
- Найпоширеніші он-лайн карти та он-лайн навігатор - Google Maps.
- Найпопулярніша мережа он-лайн маркетингу Google ads.
- Найпоширеніший браузер - Google Chrome.
- Найпоширеніша служба електронної пошти – Gmail.
- Другий за популярністю офісний пакет – Gsuite.
- Найпопулярніше он-лайн сховище приватних файлів Google Drive.
- Найпоширеніша операційна система для мобільних пристроїв - Android OS, до якої входять інші сервіси, які також є лідерами ринку у своїй галузі. Такі сервіси як:
    - Платіжна система Google Pay.
    - Магазин додатків Google Play Market.
    - Сервіс прослуховування музики Google Play Music.
    - Сервіс зі зберігання фотографій Google Photos.

Для підтримки роботи усіх сервісів на належному рівні, адже напевно жодна інша компанія не опрацьовує настільки ж великі об'єми трафіку, компанії Alphabet необхідно було мати власні центри з обробки даних. Тож маючи досвід у галузі хмарних обчислень для власних потреб, компанія Alphabet вирішила продавати власні рішення для інших компаній, тож у 2008 році було засновано Google Cloud Platform. Ця платформа пропонує хмарні рішення, як є ідентичними тим, які Alphabet використовує у своїх флагманських продуктах.

Структура Google Cloud Platform схожа на структура Amazon Web Services, а саме має такі спільні елементи:

- Мають дата центри по всьому світу (у 28 регіонах) (див. рис. 1.6).

- Регіони поділені на зони (загалом 85 зон).
- Має мережу CDN (Content Delivery Network).
- У центрах з обробки даних знаходяться великі обчислювальні сервери, потужності яких виділяються під користувача на логічному рівні, а не на фізичному.

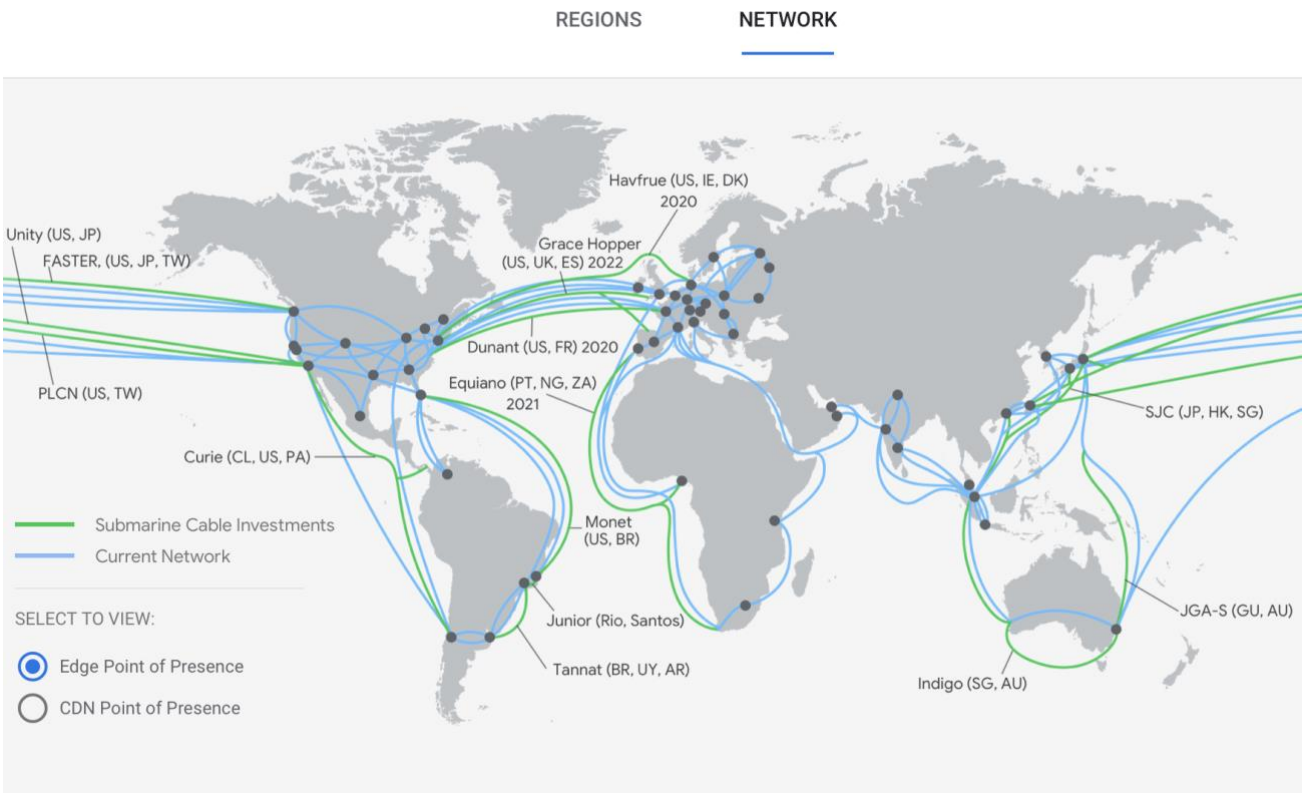

Рисунок 1.6 - Основні локації розташування дата центрів Google Cloud Platform нанесені на мапу світу. [15]

Більшість сервісів, аналоги яких є найпопулярнішими для Amazon Web Services є також і найпопулярнішими у Google Cloud Platform. Однак найпоширенішим способом використання Google Cloud Platform є задачі, пов’язані із Обробкою та аналізом великих даних. Тож розглянемо найпопулярніші сервіси з обробки даних, які надає Google Cloud Platform:

1. Google Big Table. Даний сервіс надає змогу зберігати дані у стислому вигляді, та мати високопродуктивний доступ до них. Це нереляційна база даних, її можна визначити як розріджену.

2. Google Big Query. Даний сервіс втілює без серверну концепцію, є широко масштабованим, та витрато-ефективним сервісом обробки даних. Він дозволяє отримувати доступ до даних, які містяться у таблицях за допомогою SQL запитів. Доступ може здійснюватися через інтерфейс командного рядку, графічний веб інтерфейс та за допомогою API запитів. В більшості випадків запити здійснюються до внутрішніх даних, але існує можливість робити запити і до даних, які знаходяться поза межею проекту користувача Google Cloud Platform, проте вони звісно є значно повільнішими. Користувач платить лише за об'єм опрацьованих даних.
3. Cloud Composer. Це сервіс, який повністю підтримується і обслуговується Google Cloud Platform і надає доступ до варіації Apache Airflow, який дозволяє будувати ETL процеси – процеси із вивантаження, перероблення та завантаження даних.

У наступних підрозділах проведемо порівняння аналогічних сервісів від описаних вище двох провайдерів хмарних послуг – Amazon Web Services та Google cloud Platform.

1.3.3 Інші провайдери послуг хмарних обчислень

Другим найбільшим гравцем послуг хмарних обчислень є Microsoft Azure. Перевагами цієї платформи є повсякчасна підтримка програмних продуктів компанії Microsoft, таких як операційна система Windows та мова програмування C# та її фреймворк .NET. Оскільки з точки зору ціни продуктів, а також кількості релевантних для наших задач сервісів Microsoft Azure поступається Amazon Web Services та Google cloud Platform, далі його розглядати не будемо.

Іншим помітним учасником ринку є Alibaba Cloud від китайської компанії Alibaba Group. У порівнянні з іншими постачальниками послуг хмарних обчислень, Alibaba Cloud має незрівнянно нижчу ціну на аналогічні продукти.

Проте величезним недоліком з точки зору безпеки є те, що оскільки компанія походить з Китаю – за потреби китайські спец служби або конкуренти зможуть отримати доступ до даних, які зберігаються у вашому хмарному сховищі. Також не варто забувати про напружені торгівельні стосунки із країнами заходу, які можуть накласти санкції на дану компанію.

Не останнім фактором є розташування дата центрів компанії (див. рис. 1.7).

Рисунок 1.7 - Основні локації розташування дата центрів Alibaba Cloud нанесені на мапу світу. [16]

Як видно з Рисунка 1.7, центри з обробки даних знаходяться в основному в Китаї, або поблизу в Азійському регіоні. У порівнянні з вище згаданими Amazon Web Services та Google cloud Platform, Alibaba Cloud значно менше центрів з обробки даних по всьому світу, що відчутно зменшує комфорт та спроможність користувачів, а також має меншу надійність. Тож, не зважаючи на помірну ціну, краще не варто використовувати цю платформу.

1.4 Модифікатори доступу IAM Role

Identity and Access Management (IAM) Role – це сукупність технологій, практик, підходів та спеціальних програмних засобів, які використовуються для того, щоб кожен користувач мав правильний набір прав доступу у системі, які необхідні йому для роботи. Цей підхід дає змогу дуже гнучко контролювати права доступу таким чином, щоб користувач не мав доступу до компонентів невластивих йому. Завдяки цьому підхід Identity and Access Management чудово зарекомендував себе з точки зору кібербезпеки та став стандартом для більшості хмарних обчислювальних сервісів.

1.4.1. Identity and Access Management у Amazon Web Services

Сервіс AWS Identity and Access Management (IAM) надає можливості безпечного керування доступом до сервісів та ресурсів Amazon Web Services. Використовуючи IAM, можна створювати користувачів Amazon Web Services та групи, керувати ними, а також використовувати дозволи, щоб надавати або забороняти доступ до ресурсів Amazon Web Services.

IAM – це послуга облікового запису Amazon Web Services, яка надається без додаткової оплати. Плата стягується лише за використання інших сервісів AWS, створених користувачами. Якщо ви вже зареєстровані в Amazon Web Services, щоб розпочати роботу з IAM, увійдіть у Консоль керування Amazon Web Services і почніть роботу за допомогою розділу документації.

За допомогою IAM користувачі можуть керувати доступом до API сервісів Amazon Web Services та конкретних ресурсів. IAM також дозволяє додавати особливі умови, при дотриманні яких користувач зможе використовувати AWS,

наприклад, час доби, вихідна IP-адреса, можливість використання протоколу SSL або необхідність використання пристрою багатофакторної автентифікації.

За допомогою IAM можна аналізувати права доступу до сервісів середовища Amazon Web Services. Команди з безпеки та адміністратори можуть швидко перевірити, що публічний та кросаккаунтний доступ до ваших ресурсів отримають лише користувачі з відповідними правами. Також можна легко визначити та уточнити свої правила, щоб вони дозволяли доступ лише до активних сервісів. Завдяки таким функціям буде простіше дотримуватися принципу мінімальних привілеїв.

Сервіс IAM може надати користувачам та програмам федеративний доступ до Консолі управління та API сервісів AWS з використанням існуючих систем ідентифікації, таких як Microsoft Active Directory. Для цього можна використовувати будь-яке рішення для керування посвідченнями, яке підтримує стандарт SAML 2.0, або вибрати будь-який з наших зразків федерації (єдиний вхід (SSO) у Консоль керування AWS або федерацію API).

Можливості, які надає Amazon Web Services IAM:

- Керування користувачами IAM та їх правами доступу. З Amazon Web Services IAM можна створювати користувачів, призначати їм безпечні індивідуальні дані для доступу (такі як ключі доступу, паролі та пристрої багатофакторної автентифікації) або вимагати тимчасові дані для доступу користувачів до сервісів та ресурсів Amazon Web Services. За допомогою дозволів можна керувати можливістю користувача виконувати певні дії.
- Управління ролями IAM та пов'язаними дозволами. Використовуючи IAM, можна створювати ролі та призначати дозволи, які визначають, які дії зможе виконувати сутність або сервіс AWS, яким присвоєно цю роль. Можна також задати, якій сутності можна надавати цю роль. Крім того, можна використовувати пов'язані з сервісом ролі для делегування дозволів сервісам AWS, які створюють ресурси AWS та керують ними від імені користувача.
- Управління федеративними користувачами та їх дозволами. За допомогою федерації посвідчень можна дозволити існуючим посвідченням

(користувачам, групам та ролям) в рамках організації використовувати Консоль управління AWS, викликати API AWS та отримувати доступ до ресурсів, не створюючи користувача IAM для кожного посвідчення. Використовуйте будь-яке рішення для керування посвідченнями, яке підтримує стандарт SAML 2.0, або будь-який з наших зразків федерації (єдиний вхід (SSO) у Консоль керування AWS або федерацію API).

На рисунку 1.8 розглянемо структуру IAM ролі.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ConsoleAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetAccountPublicAccessBlock",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicyStatus",
                "s3:GetBucketPublicAccessBlock",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": ["arn:aws:s3:::bucket-name"]
        },
        {
            "Sid": "AllObjectActions",
            "Effect": "Allow",
            "Action": "s3:*Object",
            "Resource": ["arn:aws:s3:::bucket-name/*"]
        }
    ]
}
```

Рисунок 1.8 - Приклад IAM ролі в Amazon Web Services.

На рисунку 1.8 зображено приклад AWS IAM ролі у форматі JSON. Розглянемо ключі полів [17]:

- Version – Елемент політики версії визначає правила синтаксису мови, які мають використовуватися для обробки політики. Щоб використовувати всі доступні функції політики, включіть наступний елемент Version перед елементом Statement у всі свої політики. IAM підтримує два значення елемента версії:
    1. 2012-10-17. Це поточна версія мови політики, і ви завжди повинні включати елемент Version і встановити для нього значення 2012-10-17. Інакше ви не можете використовувати такі функції, як змінні політики, які були виведені з цієї версії.
    2. 2008-10-17. Це була попередня версія політики. Ви можете побачити цю версію в старих існуючих політиках. Не використовуйте цю версію для будь-яких нових політик або під час оновлення існуючих правил. Новіші функції, такі як змінні політики, не працюватимуть з вашою політикою. Наприклад, такі змінні, як ${aws:username}, не розпізнаються як змінні, а натомість у політиці розглядаються як рядки літер.
- Statement - Елемент Statement є основним елементом політики. Цей елемент обов’язковий. Елемент Statement може містити один оператор або масив окремих операторів. Кожен окремий блок операторів має бути укладений у фігурні дужки { }. Для кількох операторів масив має бути узятий у квадратні дужки [ ].
- Sid - Ви можете вказати необов’язковий ідентифікатор Sid (ідентифікатор оператора) для Statement політики. Ви можете призначити значення Sid кожному оператору в масиві операторів. У службах, які дозволяють вказувати елемент ідентифікатора, наприклад SQS і SNS, значення Sid є лише субідентифікатором ідентифікатора документа політики. У IAM значення Sid має бути унікальним у політиці JSON. Елемент Sid підтримує ASCII великі

літери (A-Z), малі літери (a-z) і цифри (0-9). IAM не повертає Sid в IAM API. Ви не можете отримати певний Statement на основі цього ідентифікатора.

- Effect - Елемент Effect є обов'язковим і вказує, чи призведе оператор до дозволу чи явного відхилення. Дійсними значеннями для Ефекту є Дозволити (Allow) та Заборонити (Deny). За замовчуванням доступ до ресурсів заборонений. Щоб дозволити доступ до ресурсу, потрібно встановити для елемента Effect значення Дозволити (Allow). Щоб перевизначити дозвіл (наприклад, щоб замінити дозвіл, який в іншому випадку діє), ви встановлюєте для елемента Effect значення Deny.
- Action - Елемент Action описує конкретну дію або дії, які будуть дозволені або відхилені. Операції повинні містити елемент Action або NotAction. Кожний сервіс AWS має власний набір дій, які описують завдання, які можна виконувати за допомогою цієї служби. Наприклад, список дій для Amazon S3 можна знайти в посібнику користувача Amazon Simple Storage Service, список дій для Amazon EC2 можна знайти в довіднику API Amazon EC2, а список дій для AWS Identity and Access Management можна знайти в довідці IAM API Reference. Щоб знайти список дій для інших служб, перегляньте довідкову документацію API для служби. Ви вказуєте значення, використовуючи простір імен служби як префікс дії (iam, ec2, sqs, sns, s3 тощо), а потім назву дії, яку потрібно дозволити чи заборонити. Ім'я має відповідати дії, яку підтримує служба. Префікс і назва дії не чутливі до регістру. Наприклад, iam:ListAccessKeys те саме, що IAM:listaccesskeys. Ви можете використовувати символ підстановки (*), щоб надати доступ до всіх дій, які пропонує конкретний продукт AWS. Наприклад, наступний елемент Action застосовується до всіх дій S3. Деякі сервіси дозволяють обмежувати доступні дії. Наприклад, Amazon SQS дозволяє зробити доступним лише підмножину всіх можливих дій Amazon SQS. У цьому випадку символ * не дозволяє повністю контролювати чергу; він дозволяє лише підмножину дій, якими ви поділилися.

- Resource - Елемент Resource визначає об'єкт або об'єкти, які охоплює оператор. Statements повинні включати елемент Resource або NotResource. Ви вказуєте ресурс за допомогою ARN. Кожен сервіс має свій набір ресурсів. Хоча ви завжди використовуєте ARN для визначення ресурсу, деталі ARN для ресурсу залежать від сервіса та ресурсу. Щоб отримати інформацію про те, як вказати ресурс, зверніться до документації сервісу, для якого потрібно написати Statement. Ви можете використовувати символи підстановки як частину ресурсу ARN. Ви можете використовувати символи підстановки (* і ?) у будь-якому сегменті ARN (частини, розділені двокрапками). Зірочка (*) позначає будь-яку комбінацію символів, а знак питання (?) — будь-який окремий символ. Ви можете використовувати кілька * або ? символів у кожному сегменті.

Більшість ресурсів мають дружнє ім’я, наприклад, користувач на ім’я Боб або група користувачів на ім’я DevOps. Однак мова політики дозволів вимагає, щоб ви вказали ресурс або ресурси, використовуючи такий формат імені ресурсу Amazon (ARN) (див. рис. 1.9).

```
arn:partition:service:region:account:resource
```

Рисунок 1.9 - Шаблон побудови ARN в Amazon Web Services.

Розглянемо основні компоненти ARN (див. рис. 1.10):

- partition - визначає розділ для ресурсу. Для стандартних регіонів AWS розділом є aws. Якщо у вас є ресурси в інших розділах, розділ має назву aws-partitionname. Наприклад, розділ ресурсів у регіоні Китаю (Пекін) — aws-cn. Не можна делегувати доступ між обліковими записами в різних partitions.
- service - ідентифікує продукт AWS. Ресурси IAM завжди використовують IAM.
- region - визначає регіон ресурсу. Для ресурсів IAM це завжди залишається порожнім.

- account - визначає ідентифікатор облікового запису AWS без дефісів.
- resource - визначає конкретний ресурс за назвою.

```
arn:aws:iam::account:root
arn:aws:iam::account:user/user-name-with-path
arn:aws:iam::account:group/group-name-with-path
arn:aws:iam::account:role/role-name-with-path
arn:aws:iam::account:policy/policy-name-with-path
arn:aws:iam::account:instance-profile/instance-profile-name-with-path
arn:aws:sts::account:federated-user/user-name
arn:aws:sts::account:assumed-role/role-name/role-session-name
arn:aws:iam::account:mfa/virtual-device-name-with-path
arn:aws:iam::account:u2f/u2f-token-id
arn:aws:iam::account:server-certificate/certificate-name-with-path
arn:aws:iam::account:saml-provider/provider-name
arn:aws:iam::account:oidc-provider/provider-name
```

Рисунок 1.10 - Шаблон побудови ARN для IAM ресурсі в Amazon Web Services.

Шляхи ARN не можна створювати або керувати з Консолі керування AWS. Щоб використовувати шляхи, ви повинні працювати з ресурсом за допомогою AWS API, AWS CLI або інструментів для Windows PowerShell.

На кожен запит AWS звіряє користувача, який зробив запит з його IAM роллю і визначає, чи може користувач виконувати даний запит (див. рис. 1.11).

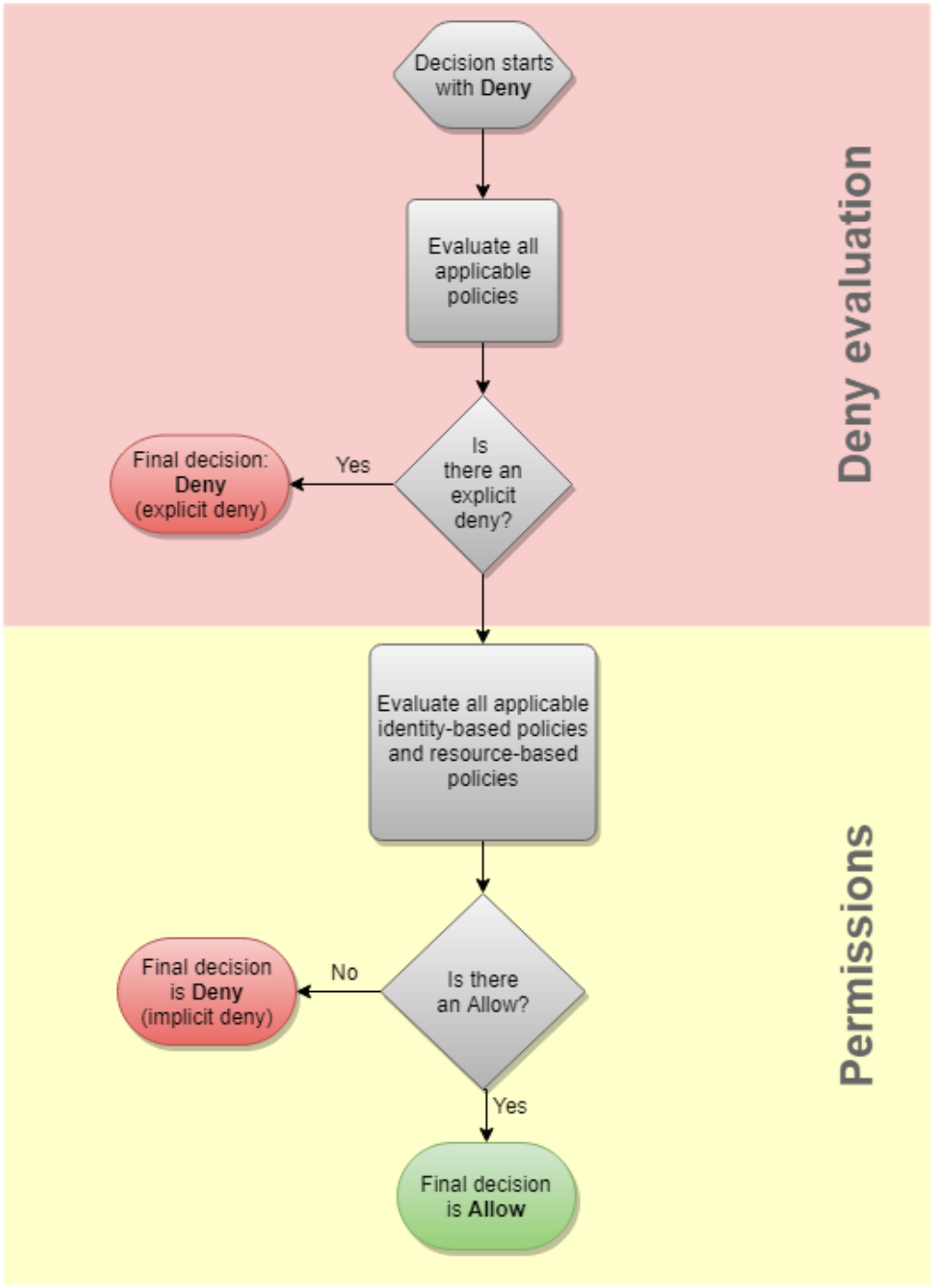

Рисунок 1.11 Схема прийняття рішення IAM сервісу на запит користувача в Amazon Web Services [18].

1.5 Віртуальна приватна хмара.

Віртуальна приватна хмара (VPC – Virtual Privat Cloud) – це попередньо конфігурований пул обчислювальних ресурсів, які виділяються у загальному для

всіх користувачів хмарному середовищі, що дає змогу забезпечити відокремленість між користувачами, що використовують ці ресурси. Тобто можна сказати, що віртуальна приватна хмара втілює модель запуску та розгортання хмарної інфраструктури, сервіси та обчислення у якій відбуваються у виділеному відокремленому середовищі.

Розмежування між різними віртуальними приватними хмарами, як правило, здійснюється за допомогою виділення на певну хмару приватної IP підмережі та певної віртуальної конструкції зв'язку, такої як VLAN, або ж набір певний набір зашифрованих каналів зв'язку [19].

Окрім механізмів з ізоляції віртуальної приватної хмари, також існують механізми доступу до цієї хмари, такі як Virtual Privat Network (VPN). За допомогою автентифікації, а також шифрування, технологія Virtual Privat Network забезпечує віддалений доступ користувача до своїх ресурсів у віртуальній приватній хмарі.

1.6 Віртуальні сервери

Віртуальні сервери є основним сервісом усіх провайдерів хмарних обчислювальних сервісів. В основі віртуальних серверів лежить концепція віртуальної машини.

Віртуальна машина – це програма та/або апаратна система, яка робить емуляцію апаратного забезпечення певної платформи; це модель обчислювальної машини, яка створена за допомогою віртуалізації обчислювальних ресурсів, таких як оперативної пам'яті, процесора, пристарою зберігання інформації (диску), а також пристроїв вводу та виводу даних.

На основі віртуальних серверів хмарні обчислювальні сервіси запускають інші свої сервіси. Тобто сервіси використовують не обчислювальні можливості напряму, а спочатку запускається віртуальний сервер, а вже на ньому запускають

сервіс, який використовує користувач. В більшості випадків сам користувач не має доступу до віртуального сервера, який використовується в основі певного хмарного сервісу, окрім випадків коли сам сервіс і є сервісом з надання послуг хмарних серверів.

Розглянемо сервіси надання віртуальних серверів у відомих нам хмарних обчислювальних платформах [20].

1.7 Безсерверні обчислення за допомогою Amazon Lambda

AWS Lambda – це сервіс безсерверних обчислень, який запускає програмний код у відповідь на певні події та відповідає за автоматичне виділення необхідних обчислювальних ресурсів [21]. Перелік подій включає зміни в стані або оновлення, наприклад, коли користувач поміщає товар у кошик на веб-сайті інтернет-комерції. AWS Lambda можна використовувати для розширення можливостей інших сервісів AWS за допомогою спеціальної логіки або для створення власних серверних сервісів із застосуванням можливостей масштабування, продуктивності та безпеки AWS. AWS Lambda автоматично запускає програмний код у відповідь на різні події, такі як HTTP-запити через Amazon API Gateway, зміна об'єктів у кошиках Amazon Simple Storage Service (Amazon S3), оновлення таблиць у Amazon DynamoDB або зміна станів у AWS Step Functions [22].

На рисунку 1.12 розглянемо застосування Amazon Lambda для автентифікації вхідних запитів.

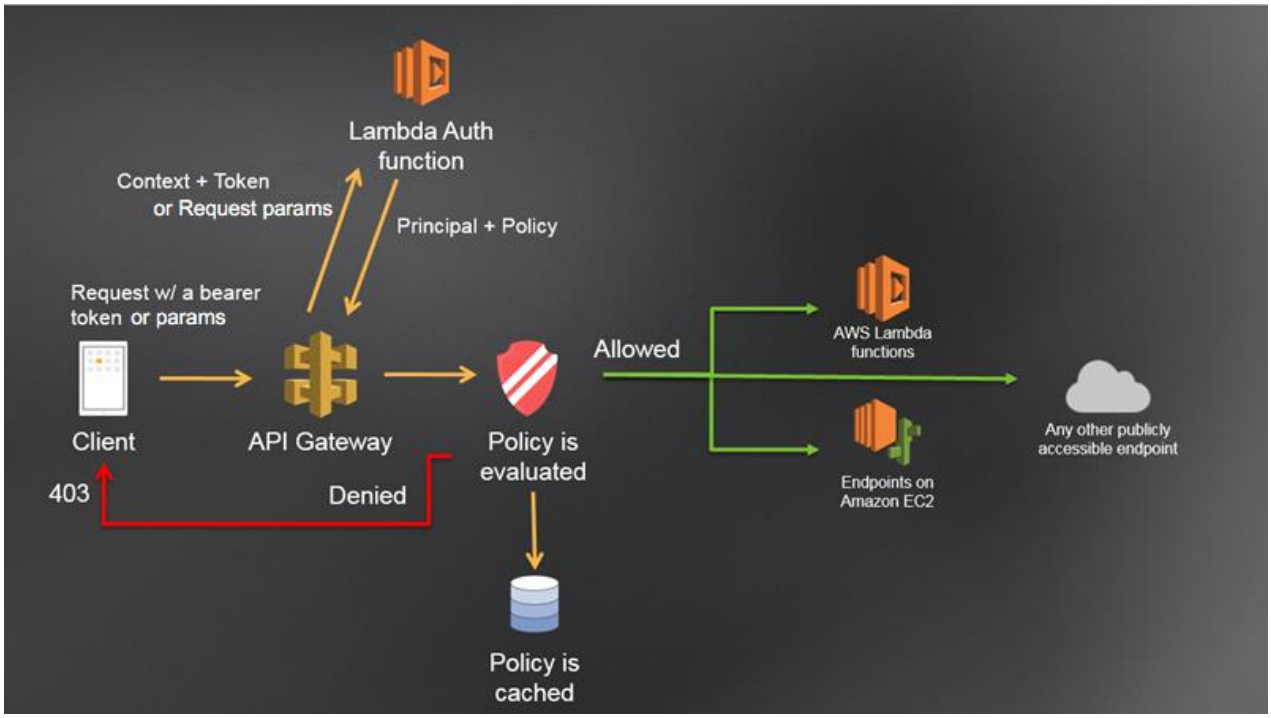

Рисунок 1.12 - Приклад схеми автентифікації вхідних запитів за допомогою сервісу безсерверних обчислень AWS Lambda.

На рисунку 1.12 продемонстровано варіацію схеми автентифікації вхідних запитів, в якій клієнт надсилає запит на ендпоінт AWS API Gateway, звідки він потрапляє до AWS Lambda, де відбувається верифікація і після чого він потрапляє в систему [23].

1.8 Висновки до розділу 1

У даному розділі розглянуто поняття акції та фондового ринку. Також розглянуто сучасні підходи до аналізу індексу акцій, а саме фундаментальний аналіз, технічний аналіз, метод експертних оцінок, а також статистичні методи. Найбільш широке застосування мають фундаментальний аналіз, технічний аналіз, тому їх було описано більш детально. Фундаментальний аналіз базується на аналізі фінансової звітності компанії, подіях на фондовому ринку, макроекономічних показниках, виробничих показниках компанії. Технічний же

аналіз навпаки який базується лише на обробці даних історичних змін ціни на цінні папери.

У другому розділі було розглянуто поняття хмарних обчислювальних сервісів, а також проведено дослідження сучасного стану ринку провайдерів хмарних обчислювальних сервісів. Було розглянуто структуру організації найбільших провайдерів хмарних обчислювальних сервісів, а саме Amazon Web Services, Google Cloud Platform, а також інших великих провайдерів.

Було розглянуто основні компоненти платформ хмарних обчислювальних сервісів, а саме:

- модифікатори доступу IAM Role;
- віртуальна приватна хмара;
- віртуальні сервери;
- безсерверні обчислення.

На основі розглянутих сервісів спробуємо спроектувати систему для прогнозування індексу акцій у реальному часі.

РОЗДІЛ 2 РОЗРОБКА МОДЕЛІ СИСТЕМИ ПРОГНОЗУВАННЯ ІНДЕКСУ АКЦІЙ В РЕАЛЬНОМУ ЧАСІ НА БАЗІ ХМАРНИХ ОБЧИСЛЮВАЛЬНИХ

2.1 Вступ до розділу 2

У цьому розділі описана розробка моделі системи прогнозування індексу акцій в реальному часі на базі хмарних обчислювальних сервісів.

Для реалізації системи необхідно обрати провайдера хмарних обчислювальних сервісів. Перед початком розробки системи, я вирішив порівняти двох провайдерів, а саме Amazon Web Services як провайдера з найбільшою часткою ринку хмарних обчислень та Google Cloud Platform, як платформу яка є чи не найпопулярнішою в задачах пов’язаних з опрацюванням великих даних та машинним навчанням.

До переваг Amazon Web Services можна віднести наступні:

- ціна на аналогічні послуги нижча ніж в Google Cloud Platform;
- має більшу кількість сервісів (більше 200, коли у Google Cloud Platform тільки 90);
- має більшу кількість користувачів що означає, що більша кількість спеціалістів з розробки мають досвід роботи з цією платформою, що полегшить пошук співробітників у майбутньому;
- надає безкоштовний пробний період на певні сервіси;
- відносно просто почати працювати;
- існують безкоштовні послуги навіть коли безкоштовний період закінчився [24].

Також розглянемо, що можна віднести до переваг Google Cloud Platform у порівнянні з Amazon Web Services:

- сервіси та рішення по роботі з великими даними вважаються більш еталонними. Такі сервіси як Google Cloud Composer, Google Bigtable, Google BigQuery, Google Cloud Dataflow [25];
- дозволяє отримати 300$ для пробного користування;

- має нативну підтримку Kubernetes, адже цю технологію розробляли інженери Google, тож сервіс який вони надають є кращим порівняно з конкурентами, проте дорожчим;
- користувацький інтерфейс є новішим (Google Cloud Platform був запущений на п’ять років пізніше за Amazon Wen Services), тож користуватись ним у багатьох випадках легше.

Хоча на перший погляд для поставленої мною задачі Google Cloud Platform видається рішенням, яке пасує найкраще, проте я обрав Amazon Web Services як платформу хмарних сервісів. В основному на це вплинуло наступне:

- в решті решт ціна на аналогічні послуги є відчутно нижчою, що в разі того, що даний проект переросте у стартап матиме неабияке значення;
- за рахунок того, що ця платформа є значно більш поширена, можна відчутно легше залучити до проекту однодумців, потенційних колег;
- також за рахунок більшої поширеності платформи можна легше знайти готові рішення, які підходять під мої задачі, що відчутно зекономить час і зусилля витрачені на розробку системи.

2.2 Розробка моделі системи прогнозування індексу акцій в реальному часі на базіхмарних обчислювальних сервісів.

Для реалізації системи прогнозування індексу акцій в реальному часі на базі хмарних обчислювальних сервісів необхідно створити обліковий запис в одному з провайдерів хмарних обчислювальних сервісів. Для даної роботи було вибрано провайдера Amazon Web Services, тож розглянемо, що необхідно для того, щоб побудувати дану систему.

Для початку необхідно створити обліковий запис на порталі [26]. Далі необхідно налаштувати IAM Policy, а саме права доступу сервісів та користувачів до ресурсів Amazon Web Services.

Після створення облікового запису Amazon Web Services та налаштування IAM Policy, необхідно створити ресурси всередині облікового запису.

2.2.1 Компоненти системи та ресурси для їх реалізації

Для подальшого розуміння схеми роботи системи, необхідно скласти список компонентів системи, їх короткий опис та вказати їх умовні позначення.

1. Amazon API Gateway. Amazon API Gateway – це повністю керований сервіс для розробників, який спрощує публікацію, обслуговування, моніторинг, захист та використання API у будь-яких масштабах. Це сервіс, який вирішує всі трудомісткі завдання щодо безпечної та надійної роботи API у будь-якому масштабі [27]. За допомогою цього сервісу створимо ендпоінт для користувацьких запитів з керування. Даний сервіс добре підходить для сервісів з малим обсягом користувачі, в разі масштабування системи необхідно буде замінити його на інший [27].
2. Amazon Simple Queue Service (SQS). Amazon Simple Queue Service (SQS) – це повністю керований сервіс черг повідомлень, за допомогою якого можна ізолювати та масштабувати мікросервіси, розподілені системи та безсерверні програми.

   За допомогою SQS можна надсилати, зберігати та отримувати повідомлення компонентів ПЗ у будь-якому масштабі без втрати повідомлень та необхідності забезпечувати доступність інших сервісів. Для даної системи буде використовуватись FIFO SQS, тобто черга у якій реалізовано принцип “First in first out” (перший зайшов перший вийшов), що дозволить зберегти послідовність команд, а за рахунок цього мінімізувати кількість помилок.

3. Amazon Lambda. AWS Lambda – це сервіс безсерверних обчислень, який запускає програмний код у відповідь на певні події та відповідає за автоматичне виділення необхідних обчислювальних ресурсів [21]. Перелік подій включає зміни в стані або оновлення, наприклад, коли користувач поміщає товар у кошик на веб-сайті інтернет-комерції. AWS Lambda можна використовувати для розширення можливостей інших сервісів AWS за допомогою спеціальної логіки або для створення власних серверних сервісів із застосуванням можливостей масштабування, продуктивності та безпеки AWS. AWS Lambda автоматично запускає програмний код у відповідь на різні події, такі як HTTP-запити через Amazon API Gateway та інші.
4. Amazon Simple Storage Service (S3). Amazon Simple Storage Service (S3) пропонує різні інструменти, які дозволяють організувати та контролювати дані для підтримки певних сценаріїв використання, скорочення витрат, забезпечення безпеки та дотримання законодавчих вимог. Дані зберігаються як об'єкти у ресурсах, які називають кошиками, при цьому розмір одного об'єкта може становити до 5 ТБ. Сховище S3 дозволяє додавати теги метаданих в об'єкти, переміщувати та зберігати дані в класах сховища S3, налаштовувати та застосовувати елементи керування доступом до даних, захищати дані від несанкціонованого використання, застосовувати аналітику великих даних, відстежувати дані на рівні об'єкта та кошика. Доступ до об'єктів можна отримати через точки доступу S3 або через ім'я вузла контейнера [28].
5. Amazon Elastic Compute Cloud (EC2). Amazon Web Services має свій сервіс з надання послуг оренди віртуальних серверів. Amazon Elastic Compute Cloud (EC2) – це істинно віртуальне середовище обчислень, яке дозволяє використовувати інтерфейси веб-сервісів для запуску серверів з різними операційними системами, завантажувати на них ваші програми, керувати дозволами на доступ до мережі та запускати образи,

використовуючи стільки систем, скільки вам потрібно [19]. Даний сервіс буде використовуватись як основа для іншого сервісу, Elastic Kubernetes Service (EKS), а саме буде виконувати функцію вузлів в Kubernetes кластері.

6. Amazon Elastic Kubernetes Service. Amazon Elastic Kubernetes Service (Amazon EKS) – це керований сервіс Kubernetes, який дозволяє легко запускати Kubernetes на AWS і в локальному середовищі. Kubernetes – це система з відкритим вихідним кодом для автоматизації розгортання та масштабування контейнерних програм, а також управління ними. Amazon EKS має сертифікат сумісності з Kubernetes, тому існуючі програми, що працюють на відкритій версії Kubernetes, сумісні з Amazon EKS.

   Amazon EKS автоматично керує доступністю та масштабованістю вузлів управління Kubernetes, які відповідають за планування контейнерів, керування доступністю додатків, зберігання даних кластера та виконання інших важливих завдань.

   Amazon EKS дозволяє вам виконувати програми Kubernetes на Amazon Elastic Compute Cloud (Amazon EC2). Сервіс Amazon EKS дозволяє використовувати переваги інфраструктури AWS: продуктивність, масштабованість, надійність та доступність, а також інтеграцію з мережею AWS та сервісами безпеки. До таких сервісів входять балансувальники Application Load Balancer для розподілу навантаження, AWS Identity and Access Management (IAM) для інтеграції з керуванням доступом на основі ролей (RBAC) та AWS Virtual Private Cloud (VPC) для підключення подів до мережі.
7. Amazon CloudWatch. Amazon CloudWatch – це сервіс моніторингу та управління, який надає цінні з практичної точки зору дані про AWS, гібридні та локальні програми та ресурси інфраструктури. CloudWatch є єдиною платформою для збору та перегляду операційних даних та даних про продуктивність у вигляді журналів та метрик. Це дозволяє

вирішити проблему розрізненості даних при моніторингу окремих систем та додатків (серверів, мереж, баз даних тощо). За допомогою CloudWatch можна відстежувати повний стек (додатки, інфраструктуру, сервіси) та використовувати попередження, журнали та події для автоматизації дій з метою скорочення середнього часу усунення проблем (Mean time to repair -MTTR). Це дозволяє звільнити важливі ресурси, щоб зосередитись на створенні додатків та підвищенні комерційної цінності.

8. Amazon Secrets Manager. AWS Secrets Manager виконує шифрування конфіденційних даних при зберіганні за допомогою ключів шифрування, що належать користувачеві і зберігаються в AWS Key Management Service (KMS). Коли ви отримуєте конфіденційні дані, Secrets Manager дешифрує їх і передає по безпечному з'єднанню TLS в локальне середовище користувача. За промовчанням Secrets Manager не записує конфіденційні дані у постійне сховище та не кешує їх. Керувати доступом до конфіденційних даних можна за допомогою точно налаштованих політик сервісу AWS Identity and Access Management (IAM) та політик на основі ресурсів. Крім того, елементам конфіденційної інформації можна надавати окремі теги та керувати доступом на основі тегів. Наприклад, конфіденційну інформацію, що використовується в робочому середовищі, можна присвоїти тег Prod, а потім прописати політику IAM для надання доступу до цієї конфіденційної інформації лише при надходженні запитів із корпоративної IT-мережі.
9. Amazon ElastiCache. Amazon ElastiCache for Redis – це дуже швидке сховище даних у пам'яті, яке забезпечує затримки в частки мілісекунди додатків, що працюють у режимі реального часу в масштабі глобальної мережі. ElastiCache для Redis створений на базі Redis з відкритим вихідним кодом, сумісний з API Redis, працює з клієнтами Redis та використовує для зберігання даних відкритий формат даних Redis.

Програми, що використовували Redis до використання даного сервісу, можуть ефективно працювати з ElastiCache для Redis без зміни коду.

10. Amazon Simple Notification Service. Amazon Simple Notification Service (Amazon SNS) – це повністю керований сервіс обміну повідомленнями для зв'язку між програмами (A2A), а також між програмами та користувачами (A2P). Функції підписки та публікації A2A надають теми з високою пропускною здатністю для push-повідомлень між розподіленими системами, мікросервісами та безсерверними програмами на основі подій за моделлю «багато хто до багатьох». Використовуючи теми Amazon SNS, системи публікацій можуть надсилати повідомлення для паралельної обробки великому числу абонентських систем, включаючи черги Amazon SQS, функції AWS Lambda та адреси HTTPS. Функціональні можливості A2P дозволяють надсилати користувачам будь-яку кількість повідомлень у вигляді SMS, мобільних push-повідомлень та повідомлень електронної пошти.

2.2.2. Схема реалізації моделі системи прогнозування індексу акцій в реальному часі на базі хмарних обчислювальних сервісів.

Розглянемо схему реалізації моделі системи прогнозування індексу акцій в реальному часі на базі хмарних обчислювальних сервісів, функції її компонентів та логіку їх взаємодії (див. рис. 2.1). Система базуватиметеся в Amazon Web Services, тож усі умовні позначення на схемі відповідають умовним позначенням із документації Amazon Web Services.

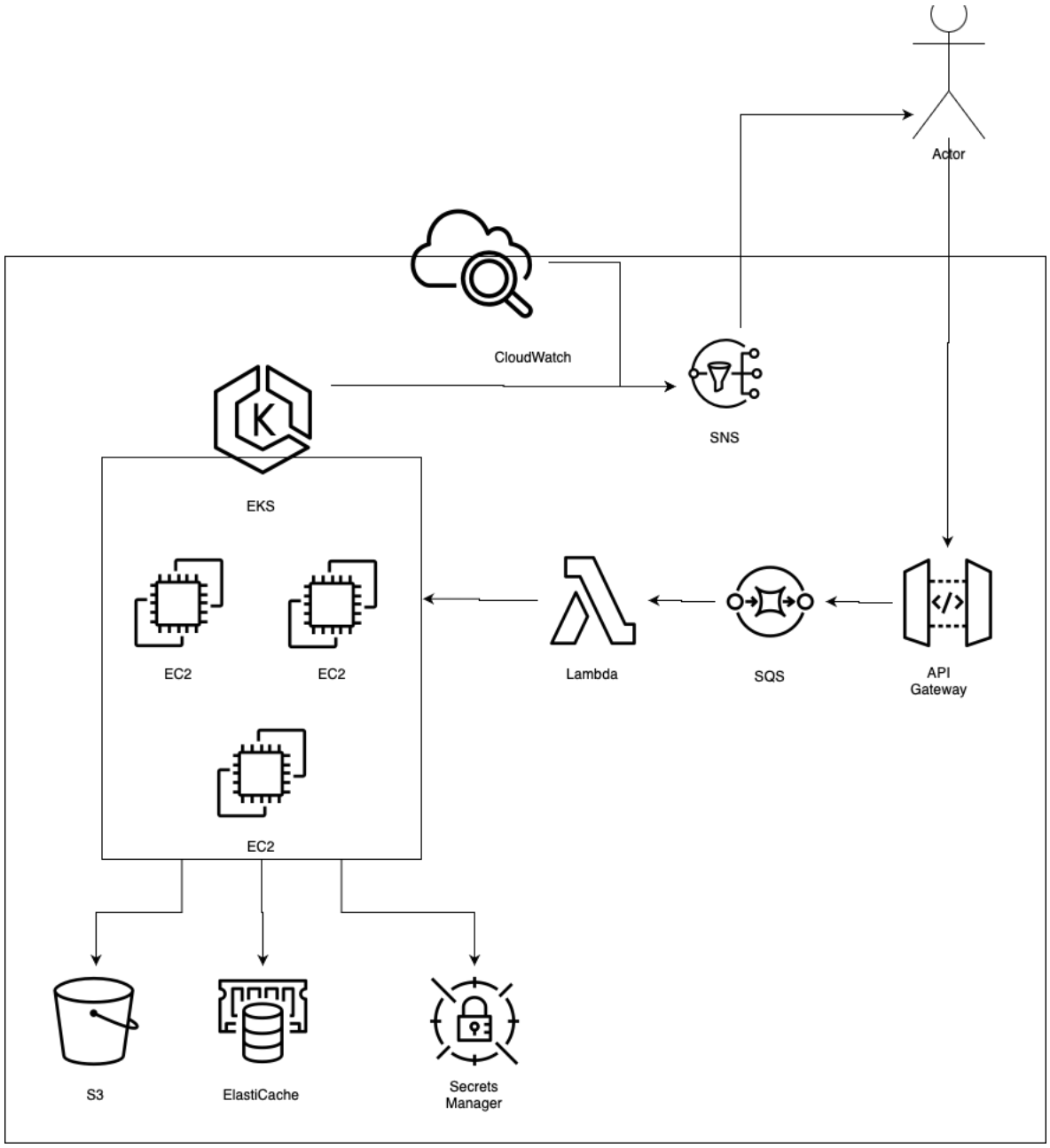

Рисунок 2.1 - Схема реалізації моделі системи прогнозування індексу акцій в реальному часі на базі хмарних обчислювальних сервісів

Розглянемо взаємодію компонентів системи:

1. Користувач за допомогою командного рядка, або графічного інтерфейсу надсилає запит на зміни до системи на ендпоінт Amazon API Gateway.
2. На рівні Amazon API Gateway налаштоване правило фільтрації кількості запитів, щоб унеможливити DDoS атаку на сервіс.

3. Від Amazon API Gateway повідомлення складається у чергу Amazon SQS, яку налаштовано працювати за принципом FIFO, що забезпечує збереження порядку надісланих повідомлень.
4. Подія наявності у Amazon SQS черзі повідомлень запускає роботу Amazon Lambda, яка робить перевірку запитів з точки зору безпеки. Це може досягатись за допомогою використання різних паролів, 2FAключів, або підтверджуючими запитами до джерела, яке має право надсилати запити.
5. Окрім того, у разі декількох одночасних запитів можливо сортувати їх за пріоритетом. До прикладу запити, які відповідають за фундаментальні зміни в системі, такі як її термінове вимкнення, будуть опрацьовуватись позачергово.
6. Потім запит направляється до Kubernetes кластера (Amazon EKS), де безпосередньо опрацьовується і вносить зміни ро роботи системи.
7. Kubernetes кластер використовуватиме безпосередньо збір даних про індекс акцій, їх обробку, насичення даних додатковою інформацією, передбачення майбутньої ціни та реалізовуватиме стратегію купівлі та продажу акцій.
8. Для зберігання великих обсягів даних, які не потребуватимуть миттєвого доступу, використовуватиметься Amazon S3 (Simple Storage Service). А саме такі дані, як давньо історичної ціни на акції, застарілу додаткову інформацію про акціє та про компанії, результати старих передбачень та інше.
9. Для зберігання даних, до яких потрібен миттєвий доступ буде використовуватись ElastiCache for Redis, що забезпечить миттєвий доступ до інформації, необхідної для роботи системи.
10. Для того, щоб Kubernetes кластер міг безпечно отримати доступ до ключів доступу до необхідних для роботи сервісів, буде використовуватись Amazon Secrets Manager, в IAM полісі якого буде прописано, що доступ до секретних значень матиме лише необхідний Kubernetes кластер.

11. Для простого поверхневого моніторингу системи буде використовуватись Amazon CloudWatch, який дозволить своєчасно виявити та проаналізувати проблеми роботи системи.
12. В разі виникнення проблеми (сервіс Amazon CloudWatch) або в разі потреби сповістити користувача про певну подію (сервіс Kubernetes) направить повідомлення до сервісу Amazon SNS, який доставить його користувачу одним з обраних методів (Email, SMS, HTTPS, Push).

Для швидкого розгортання системи буде використовуватись IaC (Infrastructure as Code) інструмент Terraform розроблений компанією HashiCorp. Це декларативний безкоштовний програмний продукт, який за допомогою стандартизованого синтаксису дозволяє відправити API запит до провайдера (Amazon Web Services) та за лічені хвилини створити усі інфраструктурні елементи з необхідними налаштуваннями. Це дозволить прописавши усі необхідні компоненти лише раз розгорнути дану інфраструктуру в будь-якому обліковому записі Amazon Web Services.

2.2.3 Компоненти Kubernetes кластера.

Kubernetes кластер – це технологія створена компанією Google для оркестрування контейнерезованих додатків, а саме їх автоматичне розгортання, координування та масштабування в межах кластера.

Розглянемо основні структурні одиниці Kubernetes кластеру:

- Вузол (Node) – сервер, який лежить в основі кластера.
- Pod – вузол, на якому запущено код додатку. На одному фізичному вузлі (node) зазвичай запущено декілька pod.
- Deployment – логічна сутність, всередині якої запускаються декілька (або багато) pod з однаковим програмним кодом, які виконують однакову

функцію. Кількість pod в одному deployment. Фізично може знаходитись на багатьох вузлах, а на одному вузлі можуть бути pod з різних deployment.

- Namespace – логічне об’єднання deployment. Саме вони і формують кластер.

Розглянемо основні компоненти Kubernetes кластера та її роль в роботі системи (див. рис. 2.2). В описі не буде згадано про технічні namespaces та deployments, а тільки ті, що відповідають за роботу логіки додатку.

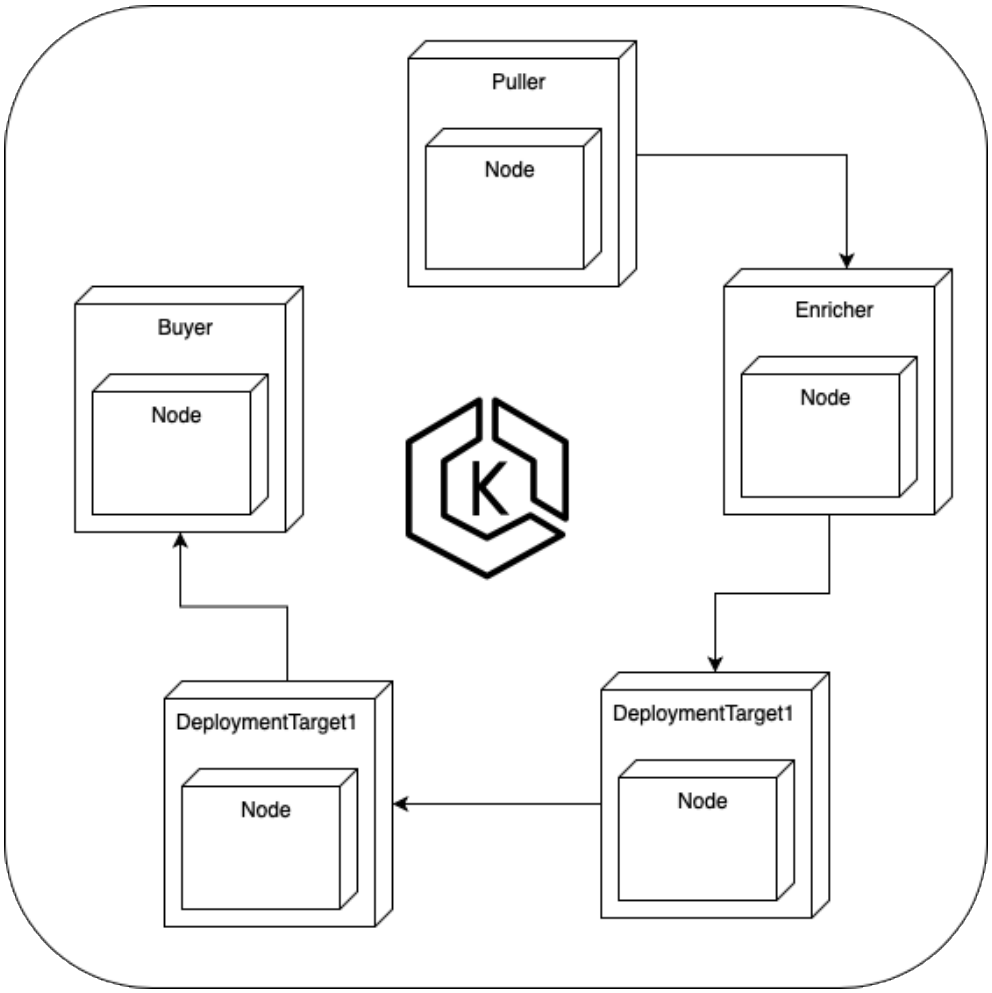

Рисунок 2.2 - Схема компонентів Kubernetes кластера

Розроблено метод прогнозування індексу акцій на базі хмарних обчислювальних сервісів (Kubernetes кластера):

1. Підключитися до біржі та завантажує дані про ціни на акції необхідних компаній (Puller).
2. Дістати різні пов’язані з компанією дані та наситити ними дані, які прийшли від Puller. Дані можуть варіюватись в залежності від обраної

компанії. До прикладу можна додавати дані аналізу певних сторінок соціальних мереж, новинних ресурсів, прогнозу погоди, тощо (Enricher).

3. Побудувати моделі та прогнози на основі даних, які надають Puller та Enricher (Forecaster).
4. Прийняти рішення спираючись на обрану стратегію, чи варто купувати акції. У разі, якщо приймає рішення, що необхідно купити певну кількість акцій – передає ці дані до Buyer (Strategist).
5. Здійснити безпосередню купівлю акцій за допомогою API одної з фондових бірж (Buyer).

Система передбачає, що користувач може віддалено вносити зміни в роботу кластера по API не втручаючись в роботу кластеру безпосередньо.

2.3 Висновки до розділу 2

В даному розділі було розроблено модель системи прогнозування індексу акцій в реальному часі на базі хмарних обчислювальних сервісів та реалізовано прототип компоненту прогнозування індексу акцій.

В результаті модель системи передбачає автономну роботу системи з можливістю віддаленого керування цією системою за допомогою вхідних API запитів. Бабуватиметься система на хмарному сервісі Amazon Web Services, та використовуватиме такі сервіси, як:

- Amazon API Gateway.
- Amazon Simple Queue Service.
- Amazon Lambda.
- Amazon Simple Storage Service.
- Amazon Elastic Compute Cloud.
- Amazon Elastic Kubernetes Service.
- Amazon CloudWatch.

- Amazon Secrets Manager.
- Amazon ElastiCache.

Система має елементи кібер безпеки, в ній передбачена можливість масштабування. Також передбачено миттєве розгортання системи в новому обліковому записі за допомогою Terraform.

Окрім проектування інфраструктурних елементів, було спроектовано структуру Kubernetes кластера, в якому знаходяться компоненти самого застосунку, такі як:

6. Puller.
7. Enricher.
8. Forecaster.
9. Strategist.
10. Buyer.

Наукова новизна полягає в розробці методу прогнозування індексу акцій на базі хмарних обчислювальних сервісів.

РОЗДІЛ 3 ПРОГНОЗУВАННЯ ІНДЕКСУ АКЦІЙ ЗА ДОПОМОГОЮ АПАРАТУ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

3.1 Вступ до розділу 3

Прототип компонента системи, який відповідає за прогнозування індексу акцій можливо було реалізувати будь-якою сучасною мовою програмування. Одними з найпривабливіших мов для цієї задачі є Javascript (Node.js), C++ та Python. Розглянемо переваги та недоліки кожної з них:

12.мова програмування C++:

1. До переваг можна віднести те, що швидкість виконання програмного коду є найбільшою у порівнянні з іншими мовами програмування.
2. Недоліком є те, що вона є значно складнішою у порівнянні з іншими вищезазначеними мовами програмування, тож розробка рішень буде відбуватись повільніше, інформації щодо розробки значно менше та знайти розробників буде значно складніше.

13.мова програмування Javascript (Node.js):

1. Безумовною перевагою є те, що JavaScript є найпоширенішою мовою програмування в світі на даний момент [29]. Це дає змогу найти багато рішень поставленої задачі, а також спрощує пошук співробітників.
2. Основними напрямками розробки на цій мові є задачі з Front-End та Back-End розробки, моделі передбачення не є профільним напрямом тож значно менше доступної інформації.

14.мова програмування Python:

1. Є найпоширенішою мовою програмування для задач з аналітики та прогнозування, безліч інформації щодо поставленої задачі, швидка розробка необхідних рішень, багато однодумців та потенційних колег.
2. Є повільнішою за інші вищеозначені мови програмування.

Зважаючи на усі вищеозначені фактори, я обрав мову програмування Python. Ключовим фактором у прийнятті рішення для мене стала швидкість, з

якою можна розробляти моделі, а також те, що для більшості випадків ця мова вважається стандартом для задач прогнозування.

Для розробки прототипу компонента системи, який відповідає за прогнозування індексу акцій необхідно визначитись з моделлю прогнозування, на якій цей компонент буде базуватись. Тож у цьому розділі будуть протестовані багатошарова нейронна мережа, згорткова нейронна мережа та мережа довгої короткої пам’яті, принципи роботи яких описані в [9].

Для реалізації компоненту системи, який відповідає за прогнозування використовувались декілька поширених варіацій нейронних мереж, а саме багатошарова нейронна мережа, згорткова нейронна мережа та мережа довгої короткої пам’яті. Для реалізації цих мереж використовувалась мова прогнозування Python та бібліотеки Pandas, Numpy, Sklearn, Keras та Tensorflow, а для візуалізації використовувалась бібліотеки Matplotlib та Seaborn. Метою даного дослідження є реалізувати декілька поширених архітектур нейронних мереж та обрати найбільш підходящу для задачі прогнозування в реальному часі, а саме ту, яка має оптимальну точність та швидко працює. Для прогнозування використовувались по хвилинні дані індексу S&P500.

3.2 Вхідні дані

Для прогнозування використовувались по хвилинні дані індексу S&P500 з 1 січня 2018 рік до 31 грудня 2018 року. Дані взяті з репозиторію на GitHub.com [30].

У цьому наборі даних присутні відомості про дату та час, індекс акції на початку хвилини, індекс на кінець хвилини, найнижча ціна впродовж хвилини, найвища ціна впродовж хвилини. Ціна наведена в доларах США. Візуалізуємо дані на рисунках 3.1 та 3.2.

0	20180101 180000	2675.000	2676.75	2674.500	2676.50
1	20180101 180100	2676.750	2677.75	2676.750	2677.75
2	20180101 180200	2677.500	2678.50	2677.500	2678.50
3	20180101 180300	2678.250	2678.25	2678.000	2678.25
4	20180101 180400	2678.130	2678.25	2678.000	2678.25
...	...	...	...	...	...
310375	20181231 160900	2508.069	2510.07	2507.770	2510.07
310376	20181231 161000	2510.070	2510.27	2509.270	2509.57
310377	20181231 161100	2509.570	2509.77	2508.769	2509.27
310378	20181231 161200	2509.270	2509.57	2509.069	2509.27
310379	20181231 161300	2509.069	2509.77	2508.770	2509.77

Рисунок 3.1 - Приклад даних індексу S&P500, які використовувались в роботі.

3.2 - Візуалізація динаміки індексу S&P500 за 2018 рік

Для торгівлі акціями в реальному часі не важливо знати точну ціну на акції, яка буде в наступну хвилину, а радше необхідно знати який буде тренд. Тож у своїх експериментах я прогнозував або індекс виросте, або індекс зменшиться через 5 хвилин після моменту прогнозування на основі даних за останні пів

години. Тобто нейронна мережа буде повертати 0, якщо ціна впаде та 1, якщо ціна виросте.

Дані ділились у співвідношенні 80 до 20, де на 80 відсотках даних нейрона мережа буде навчатись, а на 20 тестуватись. Для роботи нейронної мережі було використано останні 10000 значень.

В разі, якщо точність передбачення буде близька до 50 або менше – експеримент можна буде вважати невдалим, адже в даному випадку дане передбачення не буде відрізнятись від підкидання монетки. Проте точність не є головною метрикою в задачах бінарної класифікації. Мережа може мати 90% точності, але в разі якщо 90% значень будуть однаковими, а мережа передбачить всі 100% однаковими це не означитиме, що мережа навчилась правильно.

Для оцінки результатів бінарної класифікації було використано метрики Recall, Precision та F1-Score.

Метрика Recall демонструє відсоток дійсно позитивних значень, які мережа передбачила правильно.

$$Recall = \frac{True\ Positive}{True\ Positive\ +\ False\ Negative}$$

Метрика Precision демонструє відсоток дійсно позитивних результатів мережі до усіх позитивних, тобто якій кількості позитивних результатів можна вірити.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive\ +\ False\ Positive}$$

Метрика F1-Score є комбінованою метрикою, яка показує співвідношення між Recall та Precision. Вона є оптимальною коли прямує до 1.

$$F1 = 2 * \frac{Recall * \text{Precision}}{Recall + \text{Precision}}$$

3.3 Реалізація моделі багатошарової нейронної мережі

Було реалізовано багатошарову нейрону мережу з одним прихованим шаром, одним вхідним та одним вихідним, тобто тришарову нейрону мережу. В якості регуляризації після шарів використовувалась нормалізація батчу BatchNormalization, який нормалізуючи кожен батч (підмножину) запобігає перенавчанню.

В якості функцій активації на кожному з шарів використовувалась функція ELU, а на останньому (вихідному) шарі використовувалась функція активації Softmax з двома нейронами.

Окрім нормалізації батчу в якості регулярізації, також використовувався метод Dropout. Суть метода полягає в навчанні великої кількості схожих, але різних нейронних мереж за рахунок того, що на кожній епосі нейронна мережа забуває якийсь відсоток своїх нейронних зав’язків. Для даної нейронної мережі використовувався один компонент Dropout після першого шару з коефіцієнтом 0.7 (тобто нейрона мережа ігнорує 70% нейронів).

В якості оптимізатора використовувався Adam, який має динамічний крок навчання. Цей оптимізатор є класичним для подібних мереж та добре зарекомендував себе в подібних задачах. Початковий крок був обраний 0.002.

В якості функції витрат використовувалась крос ентропія. Для даної мережі було задано 100 епох. Наведемо архітектуру мережі на рисунку 3.3.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 dense_3 (Dense)              (None, 64)                1984

 batch_normalization_2 (Batc  (None, 64)                256
 hNormalization)

 elu_2 (ELU)                  (None, 64)                0

 dropout_1 (Dropout)          (None, 64)                0

 dense_4 (Dense)              (None, 16)                1040

 batch_normalization_3 (Batc  (None, 16)                64
 hNormalization)

 elu_3 (ELU)                  (None, 16)                0

 dense_5 (Dense)              (None, 2)                 34

 activation_1 (Activation)    (None, 2)                 0

=================================================================
Total params: 3,378
Trainable params: 3,218
Non-trainable params: 160
_________________________________________________________________
```

Рисунок 3.3 - Архітектура багатошарової нейронної мережі, використаної в експерименті

Зобразимо результати роботи мережі на рисунках 3.4 – 3.6.

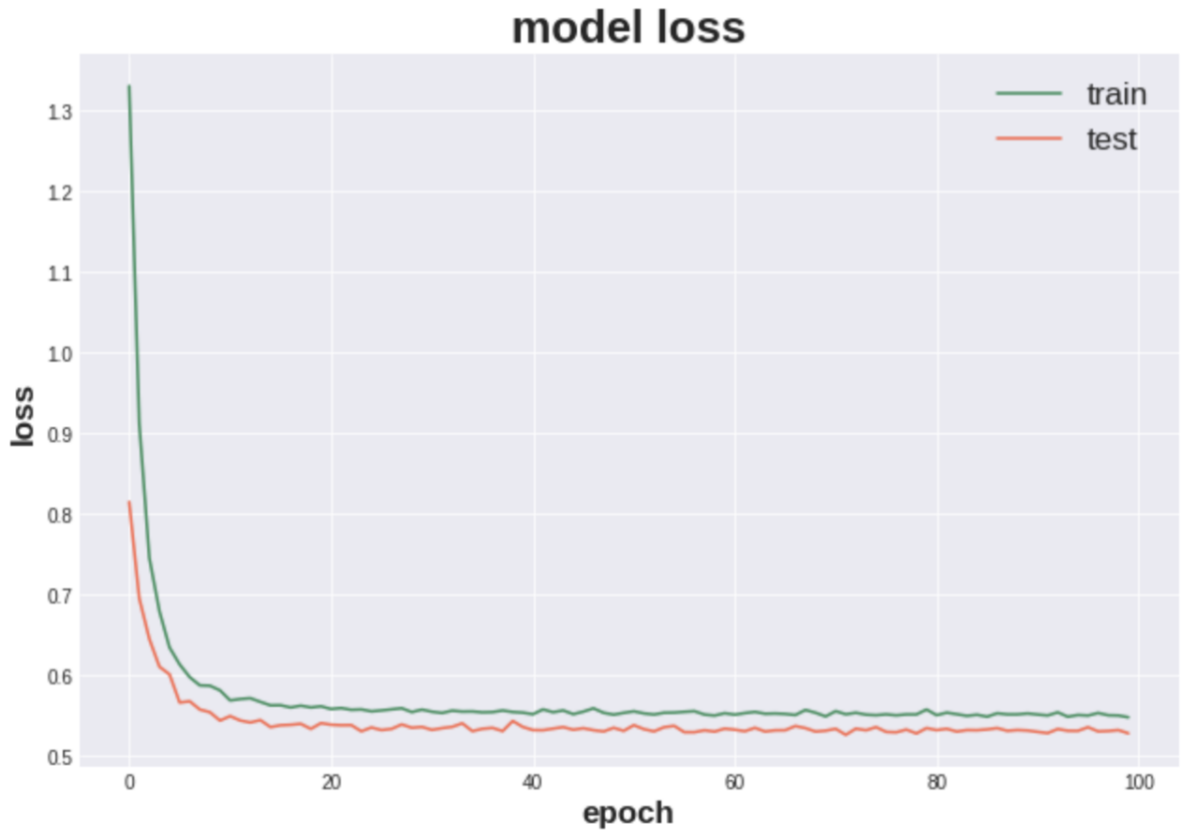

Рисунок 3.4 - Динаміка зміни похибки багатошарової нейронної мережі

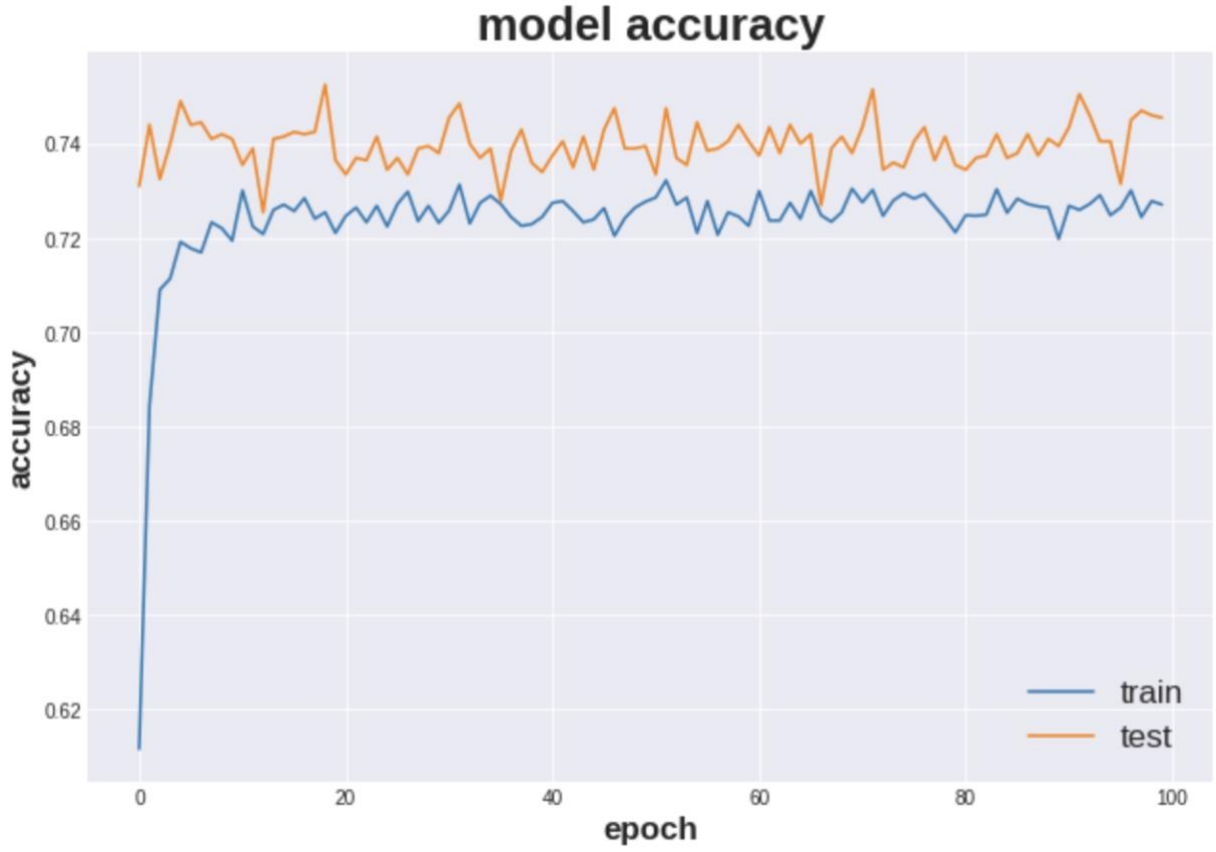

Рисунок 3.5 - Динаміка зміни точності багатошарової нейронної мережі

	precision	recall	f1-score	support
0	0.75	0.73	0.74	972
1	0.75	0.77	0.76	1021
accuracy			0.75	1993
macro avg	0.75	0.75	0.75	1993
weighted avg	0.75	0.75	0.75	1993

True positive = 0.735
False positive = 0.253
True negative = 0.768
False negative = 0.244

Рисунок 3.6 - Результати роботи багатошарової нейронної мережі

Дана архітектура показала прийнятну точність. Вона проста та швидка, та як видно з графіків, можливо понизити кількість епох для більш швидкої роботи.

3.4 Реалізація моделі згорткової нейронної мережі

Для задачі прогнозування тренду в реальному часі спробуємо використати згорткову нейронну мережу. Для її реалізації треба задати два додаткових гіперпараметри (тобто параметри, які задаються перед початеом роботи мережі та не змінюються в процесі): розмір фільтрів та їх кількість. Кількість підберають в залежності від розміру даних. Для цієї мережі було використано функцію активації LeakyReLU, а на вихідному шарі застосовано функцію сигмоїди. В якості функції витрат використовувалась крос ентропія, для Dropout було підібрано значення 0.3, а кількість епох була рівна 50.

Наведемо архітектуру мережі на рисунку 3.7.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 30, 32)            96

batch_normalization_2 (Batc  (None, 30, 32)            128
hNormalization)

elu_2 (ELU)                  (None, 30, 32)            0

dropout_1 (Dropout)          (None, 30, 32)            0

flatten_1 (Flatten)          (None, 960)               0

dense_2 (Dense)              (None, 64)                61504

batch_normalization_3 (Batc  (None, 64)                256
hNormalization)

elu_3 (ELU)                  (None, 64)                0

dense_3 (Dense)              (None, 2)                 130

activation_1 (Activation)    (None, 2)                 0

=================================================================
Total params: 62,114
Trainable params: 61,922
Non-trainable params: 192
_________________________________________________________________
```

Рисунок 3.7 - Архітектура згорткової нейронної мережі, використаної в експерименті

Зобразимо результати роботи мережі на рисунках 3.8 – 3.10.

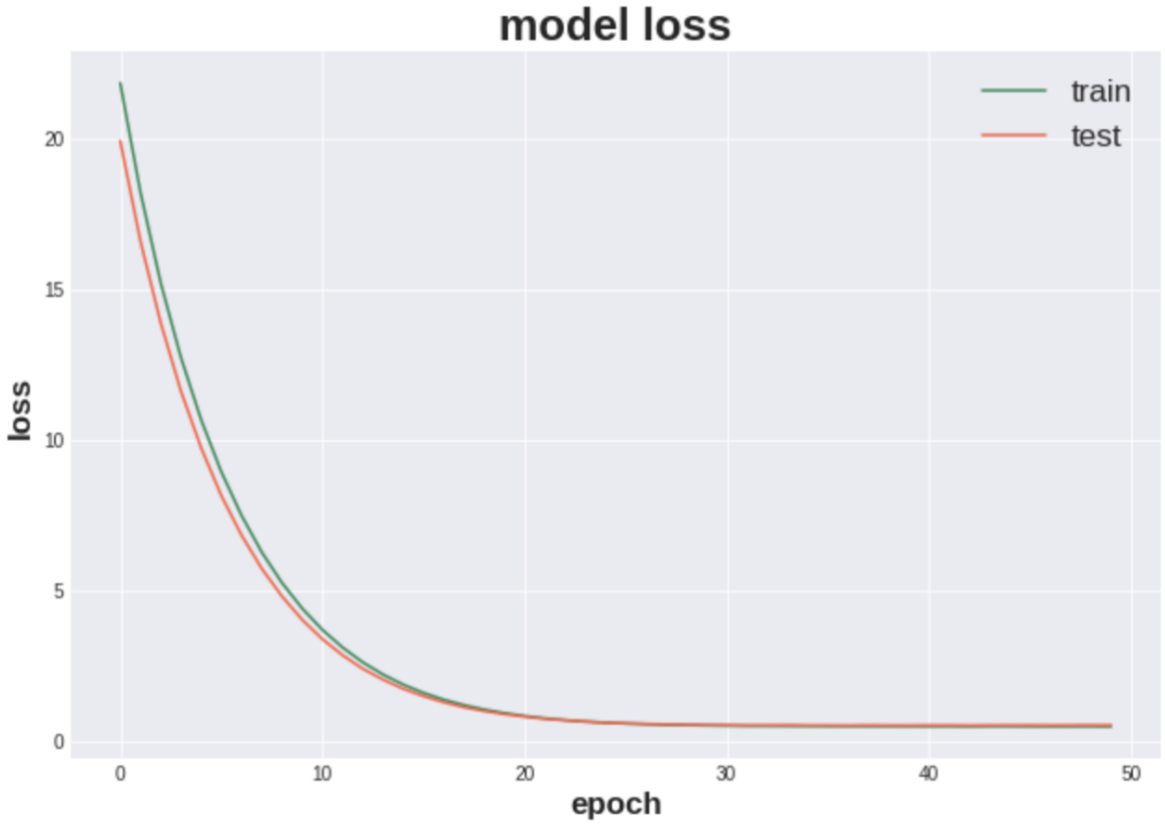

Рисунок 3.8 - Динаміка зміни похибки згорткової нейронної мережі

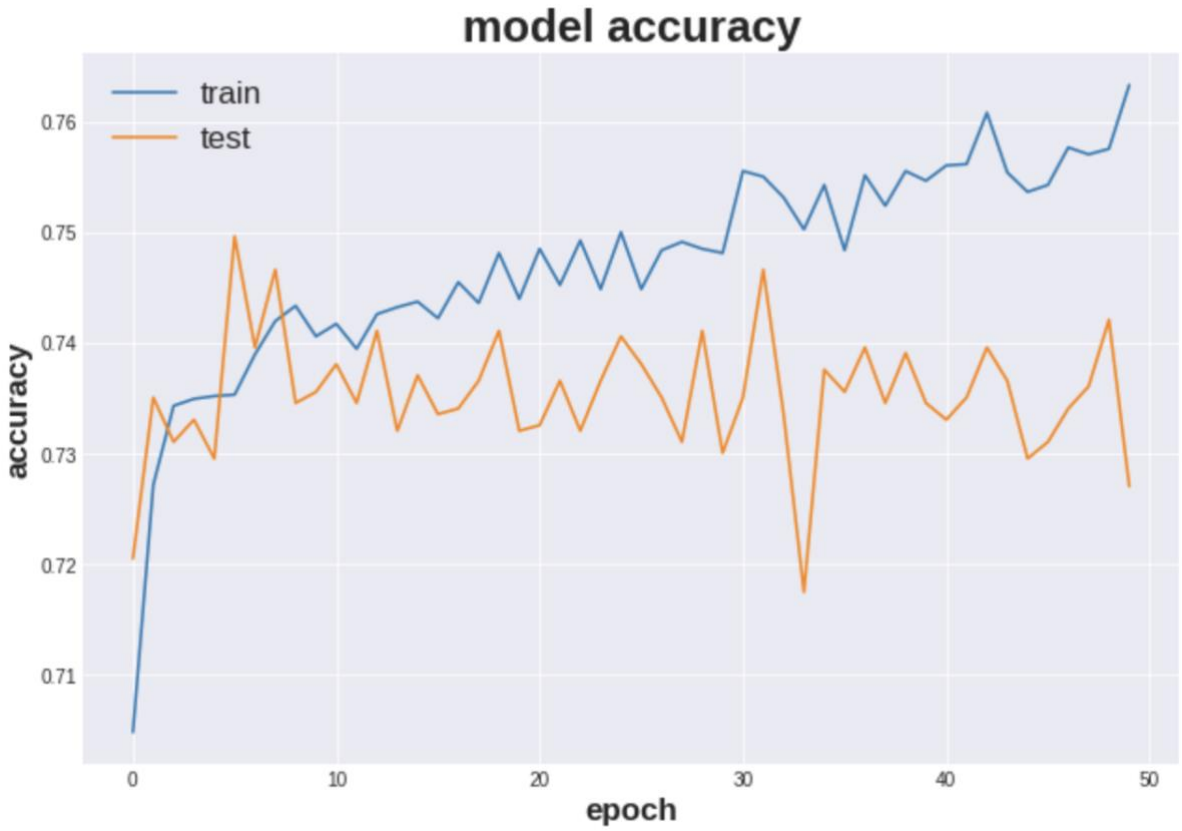

Рисунок 3.9 - Динаміка зміни точності багатошарової нейронної мережі

	precision	recall	f1-score	support
0	0.73	0.73	0.73	972
1	0.75	0.75	0.75	1021
accuracy			0.74	1993
macro avg	0.74	0.74	0.74	1993
weighted avg	0.74	0.74	0.74	1993

True positive = 0.731
False positive = 0.256
True negative = 0.747
False negative = 0.265

Рисунок 3.10 - Результати роботи згорткової нейронної мережі

Згорткова нейрона мережа показала результати співставні з простою багатоаровою, проте навчання відбувалось відчутно довше, а оскільки час грає ключову роль в торгівлі акціями в реальному часі дана модель фактично програє конкуренцію.

3.5 Реалізація моделі нейронної мережі довгої короткочасної пам’яті

Спробуємо вирішити поставлену задачу за допомогою нейронної мережі довгої короткочасної пам’яті. Вона нерідко згадується як мережа, яка добре підходить для прогнозування часових рядів, тож перевіримо чи підходить вона для задачі в реальному часі. На етапі навчання було задано 50 епох, а архітектура наведена нижче.

Наведемо архітектуру мережі на рисунку 3.11.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 30)                3840

 dropout (Dropout)           (None, 30)                0

 dense (Dense)               (None, 64)                1984

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 leaky_re_lu (LeakyReLU)     (None, 64)                0

 dense_1 (Dense)             (None, 2)                 130

 activation (Activation)     (None, 2)                 0

=================================================================
Total params: 6,210
Trainable params: 6,082
Non-trainable params: 128
```

Рисунок 3.11 - Архітектура нейронної мережі довгої короткочасної пам’яті, використаної в експерименті

Зобразимо результати роботи мережі на рисунках 3.12 – 3.14.

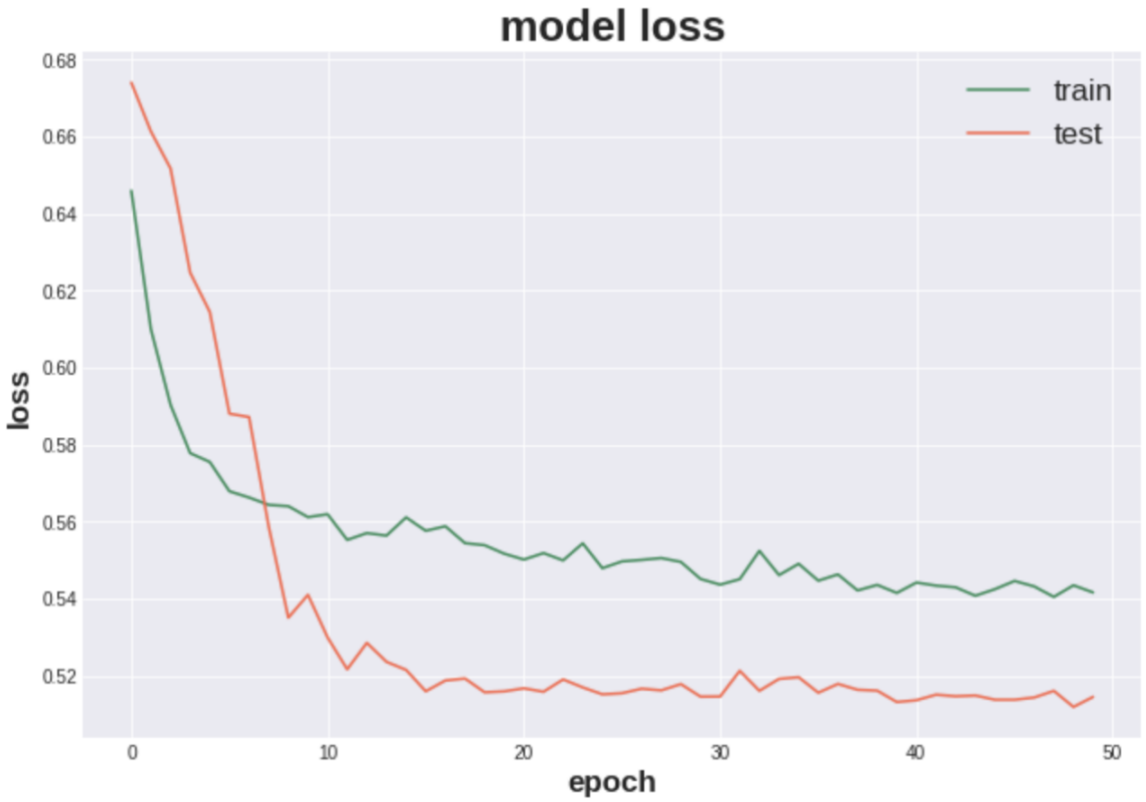

Рисунок 3.12 - Динаміка зміни похибки нейронної мережі довгої короткочасної пам’яті

Рисунок 3.13 - Динаміка зміни точності нейронної мережі довгої короткочасної пам’яті

	precision	recall	f1-score	support
0	0.74	0.72	0.73	972
1	0.74	0.76	0.75	1021
accuracy			0.74	1993
macro avg	0.74	0.74	0.74	1993
weighted avg	0.74	0.74	0.74	1993

True positive = 0.722
False positive = 0.264
True negative = 0.758
False negative = 0.254

Рисунок 3.14 - Результати роботи нейронної мережі довгої короткочасної пам’яті

Мережа показує дійсно хороші результати, які майже на тому ж рівні, що і за двох інших архітектур. Проте навчання та передбачення відбувається непорівняно довше за інші мережі. Враховуючи те, що навчання та передбачення займає більше хвилини вона не може використовуватись в прогнозуванні в реальному часі.

3.6 Висновки до розділу 3

Були побудовані багатошарова нейронна мережа, мережа довгої короткої пам’яті та згорткова нейронна мережа, які аналізували по хвилинні дані індексу S&P500. Вони передбачали, як чи зросте або впаде індекс S&P500 через 5 хвилин після моменту прогнозування на основі даних за останні 30 хвилин.

В результаті експериментів усі три архітектури нейронних мереж показали схожі гарні результати передбачення, проте показали різний час, який знадобився їм на навчання та передбачення.

Мережа довгої короткої пам’яті навчалася і передбачала загалом більше хвилини, тож вона однозначно не підходить для прогнозування в реальному часі. Загорткова нейронна мережа показала кращий час, проте вона має багато параметрів. І врешті звичайна багатошарова нейронна мережа показала найкращий час та трохи кращий результат, тож для даного компонента доцільно обрати її.

РОЗДІЛ 4 РОЗРОБКА ВЛАСНОГО СТАРТАП ПРОЄКТУ

4.1 Сутність та особливості стартапу

Мій варіант виробу повинен виконувати функції інтуїтивно зрозумілого для кінцевого користувача інтерфейсного програмного забезпечення для моделювання та прогнозування фінансових процесів з використанням методів машинного навчання та різних архітектур нейронних мереж. В таблиці 4.1 наведено основну ідею проекту.

Таблиця 4.1 – Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
Система трейдингу акціями в реальному часі на основі хмарних сервісів	Розробка стратегій керування інвестиціями	Зручна перевірка стратегій
	Перевірка гіпотез доцільності інвестування	Повсякчасна доступність
	Навчальна	Примноження інвестицій користувача

Далі необхідно визначити конкурентів на ринку і зробити порівняльний аналіз, виявити їх переваги та недоліки. Прямих аналогів не існує, тому вибрали конкурентів, частини функціоналу яких співпадає. В таблиці 4.2 наведені основні характеристики ідеї.

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	*Техніко-економічні характеристики ідеї*	*(потенційні) товари/концепції конкурентів*			*W (слабка сторона)*	*N (нейтральна сторона)*	*S (сильна сторона)*
		Мій проект	*Etoro*	*Звичайний трейдер*			
1	Лояльність споживачів	Висока в своїй області	Висока в своїй області	Висока в своїй області		+	

Продовження таблиці 4.2

2	Ціна	Задовільна	Задовільна	Висока		+	
3	Функціонал	Торгівля акціями обраних компаній, розробка стратегій	Торгівля акціями обраних компаній	Торгівля акціями обраних компаній		+	

4.2 Технологічний аудит ідеї проекту

Наступним необхідним кроком є аудит технологій. В таблиці 4.3 наведений аналіз складових частин.

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

№ п/п	*Ідея проекту*	*Технології її реалізації*	*Наявність технологій*	*Доступність технологій*
1	Система трейдингу акціями в реальному часі на основі хмарних сервісів	Використання мови програмування C++	Відсутні	Недоступні
2		Використання мови програмування JavaScript	Відсутні	Недоступні
3		Використання мови програмування Python	Наявні	Доступні
Обрана технологія реалізації ідеї проекту: мова програмування Python				

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Стартап не існує в вакуумі, тому необхідно проаналізувати ринок, на який ми виходимо. В таблиці 4.4 наведена характеристика ринку.

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	*Характеристика*
1	Кількість головних гравців, од	150
2	Загальний обсяг продаж	1 млн. $
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	10

Наступним кроком необхідно охарактеризувати основні групи потенційних користувачів продукту і скласти опис вимог кожної такої групи (таблиця 4.5).

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
1	Примноження капіталу в результаті інвестицій	Спеціалісти	Мають досвід в області	Якість
2	«Бібліотека» області	Студенти	Початківці	Цікавить інформація
3	Каталог методів	Компанії	Цікавить що працює	Ціна, якість

Далі необхідно проаналізувати загрози, які можуть виникнути та перешкодити вдалому запуску проекту. Результати представлені у таблиці 4.6.

Таблиця 4.6 – Фактори загроз

№ п/п	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1	Мала інформованість	Багато людей можуть просто не знати про мережу	Рекламна компанія
2	Складність інтерфейсу	Багато людей можуть просто не розібратись	Спростити інтерфейс

Також необхідно розглянути фактори, які можуть сприяти проекту. Результати представлені у таблиці 4.7.

Таблиця 4.7 – Фактори можливостей

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1	Інструмент для перевірки	Інструмент для перевірок ідей	Створення якісного інструменту
2	База інформації	База стратегій для різних задач	Розширення бази

Далі необхідно охарактеризувати конкурентне середовище. Результати аналізу наведено у таблиці 4.8. Також виконаємо аналіз конкуренції за моделлю п’яти сил конкуренції Майкла Портера. Результати аналізу зведено в таблицю 4.9.

Таблиця 4.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)*
1. Вказати тип конкуренції: Чиста	Багато систем/ спеціалістів	Розробити відомий продукт
2. За рівнем конкурентної боротьби: міжнаціональна	Є системи з різних країн	Розширити аудиторію
3. За галузевою ознакою: міжгалузева	Застосовується в різних цілях	Розширити можливості
4. Конкуренція за видами товарів: товарно-родова	Конкуренція з іншими подібними інструментами	Розширити функціонал

Продовження таблиці 4.8

5. За характером конкурентних переваг: Нецінова	Різні інструменти для моделювання	Збільшення якості та функціоналу
6. За інтенсивністю: Марочна	Бренд надає переваги	Розвивати бренд

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	*Прямі конкуренти в галузі*	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*
	Etoro, Трейдери	*Спеціалізованість ринку*	*Їх методи та реалізація*	*Їх роботи*	*Більш якісний*
Висновки	Інтенсивна боротьба навряд буде	Є можливості та потенційні конкуренти	Не диктують	Не диктують	Товари замінники відсутні

Результати аналізу підтверджують, що на ринку сприятливі умови для створення і запуску даного стартап-проекту. Були сформульовані та обґрунтовані перелік факторів конкурентоспроможності. Аналіз оформлено в таблицю 4.10.

Таблиця 4.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	*Фактор конкурентоспроможності*	*Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)*
1	Багатофункціональність	Проект на багато що здатен
2	Доступність програмного продукту	Загальнодоступність та кросплатфоменість
3	Обслуговування	Періодичні оновлення бази та покращення інструменту

Після проведення аналізу можна виділити сильні та слабкі (які потребують вдосконалення) сторони продукту (таблиця 4.11).

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін системи

№ п/ п	*Фактор конкурентоспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів*						
			-3	*-2*	*-1*	*0*	*+1*	*+2*	*+3*
1	Багатофункціональність	20				+			
2	Доступність програмного продукту	20				+			
3	Обслуговування	19			+				

SWOT-аналіз дозволяє оцінити можливості та загрози бізнесу, а також сильні і слабкі сторони продукту. Результати наведені у таблиці 4.12.

Таблиця 4.12 – SWOT- аналіз стартап-проекту

Сильні сторони: Якість та багатофункціональність	Слабкі сторони: Клієнтська база
Можливості: попит та вдосконалення	Загрози: конкуренція, збут

На основі SWOT-аналізу було спроектовано альтернативну ринкову поведінку для інтеграції стартап-проекту (таблиця 4.13).

Таблиця 4.13 – Альтернативи ринкового впровадження стартап проекту

№ п/п	*Альтернатива (орієнтовний комплекс заходів) ринкової поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
1	Швидкий вихід з мінімально функціональним продуктом	20%	4 місяці
2	Поступовий вихід з якісним продуктом	80%	8 місяців

4.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії розпочинається з визначення стратегії охоплення ринку. Для цього необхідно охарактеризувати цільові групи потенційних споживачів (таблиця 4.14).

Таблиця 4.14 – Вибір цільових груп потенційних споживачів

№ п/п	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1	Спеціаліст в області	Висока	70%	Низька	Середня
2	Студент	Висока	20%	Низька	Середня
3	Компанія	Висока	10%	Низька	Середня
Які цільові групи обрано:1, 2, 3					

Для роботи в обраних сегментах ринку сформуємо базову стратегію розвитку (таблиці 4.15, 4.16, 4.17).

Таблиця 4.15 – Визначення базової стратегії розвитку

№ п/п	*Обрана альтернатива розвитку проекту*	*Стратегія охоплення ринку*	*Ключові конкурентоспроможні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку**
1	3	Стратегія спеціалізації	Функціональність, багатогалузевість та якість	Стратегія диференційованого маркетингу

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	*Чи є проект «першопрохідцем» на ринку?*	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки**
1	Так	Так	Ні	Стратегія лідеру

Таблиця 4.17 – Визначення стратегії позиціонування

№ п/п	*Вимоги до товару цільової аудиторії*	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап-проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)*
1	Якість, функціональність, зрозумілий інтерфейс	Стратегія диференційованого маркетингу	Якість, висока функціональність, велика кількість інформації	Імідж, якість, спеціалізація

4.5 Розроблення маркетингової програми стартап-проекту

Кінцевим етапом є формування маркетингової концепції товару. Спочатку необхідно визначити ключових переваг концепції потенційного товару (таблиця 4.18). Також опишемо три рівні моделі товару (таблиця 4.19).

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)*
1	Моделювання стратегій	Інструментарій для моделювання	Спеціалізація
2	Пошук готових методів	Список наявних методів	Унікальність

Таблиця 4.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Моделювання стратегій та пошук готових методів		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх/Тл/Е/Ор
	Програмний продукт – складна, комплексна система		
	Якість: тестування фірмами		
	Пакування - інсталятор		
	Марка: назва організації-розробника + назва товару		
III. Товар із підкріпленням	До продажу: гнучка оплата, доставка		
	Після продажу: гарантія якості та оновлення		
За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності			

Далі необхідно визначити цінові межі продукту (таблиця 4.20).

Таблиця 4.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	1000	2000	1 млн.	500-800

4.6 Висновки до розділу 4

Був виконаний перший етап розроблення стартап проекту, а саме, маркетинговий аналіз стартап проекту.

Можна сказати, що існує можливість ринкової комерціалізації проекту, адже на ринку відсутні конкуренти та наявна величезна потенційна аудиторія, до того ж рентабельність проекту є задовільною та ринок зростає.

Для реалізації проекту краще обрати варіант, де застосовується мова програмування Python. У проекту є великі шанси для подальшого розвитку та покращення.

ВИСНОВКИ

В даній роботі проведено огляд засобів аналізу та прогнозування індексу акцій та можливості хмарних обчислювальних сервісів, а також було розроблено модель системи прогнозування індексу акцій в реальному часі на базі хмарних обчислювальних сервісів та реалізовано прототип компоненту прогнозування індексу акцій.

В результаті модель системи передбачає автономну роботу системи з можливістю віддаленого керування цією системою за допомогою вхідних API запитів. Бабуватиметься система на хмарному сервісі Amazon Web Services, та використовуватиме такі сервіси, як:

15. Amazon API Gateway.
16. Amazon Simple Queue Service.
17. Amazon Lambda.
18. Amazon Simple Storage Service.
19. Amazon Elastic Compute Cloud.
20. Amazon Elastic Kubernetes Service.
21. Amazon CloudWatch.
22. Amazon Secrets Manager.
23. Amazon ElastiCache.

Система має елементи кібер безпеки, в ній передбачена можливість масштабування. Також передбачено миттєве розгортання системи в новому обліковому записі за допомогою Terraform.

Окрім проектування інфраструктурних елементів, було спроектовано структуру Kubernetes кластера, в якому знаходяться компоненти самого застосунку, такі як:

24. Puller.
25. Enricher.

26. Forecaster.
27. Strategist.
28. Buyer.

Також було розроблено і протестовано компонента з прогнозування індексу акцій. Були побудовані багатошарова нейронна мережа, мережа довгої короткої пам’яті та згорткова нейронна мережа, які аналізували по хвилинні дані індексу S&P500. Вони передбачали, як чи зросте або впаде індекс S&P500 через 5 хвилин після моменту прогнозування на основі даних за останні 30 хвилин.

В результаті експериментів усі три архітектури нейронних мереж показали схожі гарні результати передбачення, проте показали різний час, який знадобився їм на навчання та передбачення.

Мережа довгої короткої пам’яті навчалася і передбачала загалом більше хвилини, тож вона однозначно не підходить для прогнозування в реальному часі. Загорткова нейронна мережа показала кращий час, проте вона має багато параметрів. І врешті звичайна багатошарова нейронна мережа показала найкращий час та трохи кращий результат, тож для даного компонента доцільно обрати її.

В майбутньому можна вдосконалити систему реалізувавши інші компоненти моделі, такі як Enricher, що дасть системі додаткові про компанії, акції яких вона намагається передбачити.

ПЕРЕЛІК ПОСИЛАНЬ

1. Про акціонерні товариства : Закон України № 514-VI від 17.09.2008 // Відомості Верховної Ради України. 2008. № 50–51. URL: https://zakon.rada.gov.ua/laws/show/514-17#Text
2. Про цінні папери і фондовий ринок: Закон України від 23.02.2006 р. - №3480-IV. URL: https://zakon.rada.gov.ua/laws/show/3480-15#Text
3. Бондар О. С. Прогнозування динаміки фондового ринку методами статистичного моделювання/ Білоцерківський національний аграрний університет, 2016. URL: http://rep.btsau.edu.ua/bitstream/BNAU/1441/1/Prohnozuvannia%20dynamiky%20fondovoho%20rynku.pdf
4. Ліхновський, П. М. Фундаментальний аналіз у прийнятті інвестиційних рішень на фондовому ринку/ Видавничо-поліграфічний центр Тернопільського національного економічного університету "Економічна думка", 2014. URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwj6yZDwtMz0AhV1hf0HHacUBmIQFnoECAIQAQ&url=https%3A%2F%2Fwww.econa.org.ua%2Findex.php%2Fecona%2Farticle%2Fdownload%2F595%2F358&usg=AOvVaw1ntKCsZzxIKD1JLvyfPP9R
5. Schabackers, Richard W. Stock Market Theory and Practice, 2011. URL: https://www.abebooks.co.uk/book-search/title/stock-market-theory-practice/author/richard-schabacker/
6. Романчук А.Л. Статистичні методи в системі аналізу фінансових інвестицій/ Миколаївський національний університет імені В.О. Сухомлинського. URL: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?I21DBN=LINK&P21DBN=UJRN&Z21ID=&S21REF=10&S21CNR=20&S21STN=1&S21FMT=ASP_meta&C21COM=S&2_S21P03=FILA=&2_S21STR=Vchtei_2017_1-2_18

7. Орлов А.И. Экспертные оценки. Учебное пособие. - М.: 2002. - 31 с. URL: http://www.aup.ru/books/m154/
8. Засоби прогнозування індексу акцій на основі апарата штучних нейронних мереж, 2020. URL: https://ela.kpi.ua/handle/123456789/37035
9. Amazon Leads $150-Billion Cloud Market. URL: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/
10. The 100 largest companies in the world by market capitalization in 2021, URL: https://www.statista.com/statistics/263264/top-companies-in-the-world-by-market-capitalization/
11. What are AWS availability zones. URL: https://getcodelove.com/what-are-aws-availability-zones/
12. AWS Pricing Calculator, URL: https://calculator.aws/#/
13. AWS Data Centers. URL: https://aws.amazon.com/compliance/data-center/data-centers/
14. AWS Services List - Top 10 AWS Services. URL: https://mindmajix.com/top-aws-services
15. Cloud Locations. URL: https://cloud.google.com/about/locations#network
16. Alibaba Cloud's Global Infrastructure. URL: https://www.alibabacloud.com/global-locations#J_7563247410
17. AWS Identity and Access Management Documentation. URL: https://docs.aws.amazon.com/iam/index.html
18. Policy evaluation logic. URL: https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_evaluation-logic.html
19. Amazon Virtual Private Cloud features. URL: https://aws.amazon.com/vpc/features/
20. Amazon EC2 features. URL: https://aws.amazon.com/ec2/features/

21. Serverless computing with AWS Lambda. URL: https://www.infoworld.com/article/3210726/serverless-computing-with-aws-lambda.html
22. Amazon Lambda features. URL: https://aws.amazon.com//lambda/features/
23. Use API Gateway Lambda authorizers. URL: https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-use-lambda-authorizer.html
24. Google Cloud vs AWS: Difference Between AWS and GCP. URL: https://www.guru99.com/google-cloud-vs-aws.html
25. Google Cloud Big Data: Building Your Big Data Architecture on GCP. URL: https://cloud.netapp.com/blog/gcp-cvo-blg-google-cloud-big-data-build-a-big-data-architecture-on-gcp
26. Sign up for AWS. URL: https://portal.aws.amazon.com/billing/signup#/start
27. Amazon API Gateway features. URL: https://aws.amazon.com/api-gateway/features/
28. Amazon S3 features. URL: https://aws.amazon.com/s3/features/
29. Top programming languages: Most popular and fastest growing choices for developers. URL: https://www.zdnet.com/article/top-programming-languages-most-popular-and-fastest-growing-choices-for-developers/
30. FutureSharks/financial-data. URL: https://github.com/FutureSharks/financial-data/tree/master/pyfinancialdata/data/stocks/histdata/SPXUSD

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
# multilayer neural network
import matplotlib.pylab as plt
import seaborn as sns
sns.despine()
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import BatchNormalization
#from keras.layers import Merge
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, CSVLogger, EarlyStopping
from tensorflow.keras.optimizers import RMSprop, Adam, SGD, Nadam
from keras.layers.advanced_activations import *
from tensorflow.keras.layers import Convolution1D, MaxPooling1D
from keras.layers.recurrent import LSTM, GRU
from keras import regularizers
import theano
theano.config.compute_test_value = "ignore"


def shuffle_in_unison(a, b):
    assert len(a) == len(b)
    shuffled_a = np.empty(a.shape, dtype=a.dtype)
    shuffled_b = np.empty(b.shape, dtype=b.dtype)
    permutation = np.random.permutation(len(a))
    for old_index, new_index in enumerate(permutation):
        shuffled_a[new_index] = a[old_index]
        shuffled_b[new_index] = b[old_index]
    return shuffled_a, shuffled_b


def create_Xt_Yt(X, y, percentage=0.8):
    p = int(len(X) * percentage)
    X_train = X[0:p]
    Y_train = y[0:p]

    X_train, Y_train = shuffle_in_unison(X_train, Y_train)
```

```
    X_test = X[p:]
    Y_test = y[p:]

    return X_train, X_test, Y_train, Y_test


data = pd.read_csv("data.csv")
data = data.loc[:, '2668.000000.1'].pct_change().dropna().tolist()[-10000:]

plt.style.use('seaborn-darkgrid')
plt.title('S&P 500')
plt.plot(data, color = 'green')
plt.show()

window = 30
emb_size = 1
step = 1
forecast = 5
X, Y = [], []
for i in range(0, len(data), step):
    try:
        x_i = data[i:i+window]
        y_i = data[i+window+forecast]

        last_close = x_i[window-1]
        next_close = y_i

        if last_close < next_close:
            y_i = [1, 0]
        else:
            y_i = [0, 1]

    except Exception as e:
        print(e)
        break
```

```
    X.append(x_i)
    Y.append(y_i)

X = [(np.array(x) - np.mean(x)) / np.std(x) for x in X]
X, Y = np.array(X), np.array(Y)

X_train, X_test, Y_train, Y_test = create_Xt_Yt(X, Y)

model = Sequential()

model.add(Dense(64, input_dim=30,
                activity_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(ELU())
model.add(Dropout(0.7))
model.add(Dense(16,
                activity_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(ELU())
model.add(Dense(2))
model.add(Activation('softmax'))

opt = Nadam(lr=0.002)


reduce_lr = ReduceLROnPlateau(monitor='val_acc', factor=0.9, patience=25, min_lr=0.000001, verbose=1)
checkpointer = ModelCheckpoint(filepath = "mnn.hdf5", verbose = 1, save_best_only = True)

model.compile(optimizer=opt,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

print(model.summary())

history = model.fit(X_train, Y_train,
          epochs = 100,
```

```
        batch_size = 128,
        verbose=1,
        validation_data=(X_test, Y_test),
        callbacks=[reduce_lr, checkpointer],
        shuffle=True)

model.load_weights("mnn.hdf5")
pred = model.predict(np.array(X_test))

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

C = confusion_matrix([np.argmax(y) for y in Y_test], [np.argmax(y) for y in pred])
print(classification_report([np.argmax(y) for y in Y_test], [np.argmax(y) for y in pred]))
C = C / C.astype(np.float).sum(axis=1)
print('True positive = ',  round(C[0][0], 3))
print('False positive = ',  round(C[0][1], 3))
print('True negative = ',  round(C[1][1], 3))
print('False negative = ',  round(C[1][0], 3))

plt.style.use('seaborn-darkgrid')
plt.figure(figsize = (10.5, 7))
plt.plot(history.history['loss'], color = 'seagreen')
plt.plot(history.history['val_loss'], color = 'tomato')
plt.title('model loss', fontsize=24, fontweight='bold')
plt.ylabel('loss', fontsize=17, fontweight='bold')
plt.xlabel('epoch', fontsize=17, fontweight='bold')
plt.legend(['train', 'test'], loc='best', fontsize=17)
plt.show()

plt.figure(figsize = (10.5, 7))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy', fontsize=24, fontweight='bold')
plt.ylabel('accuracy', fontsize=17, fontweight='bold')
plt.xlabel('epoch', fontsize=17, fontweight='bold')
```

```
plt.legend(['train', 'test'], loc='best', fontsize=17)
plt.show()

# Convolutional neural network

import matplotlib.pylab as plt
import seaborn as sns
sns.despine()
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import BatchNormalization
#from keras.layers import Merge
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, CSVLogger, EarlyStopping
from tensorflow.keras.optimizers import RMSprop, Adam, SGD, Nadam
from keras.layers.advanced_activations import *
from tensorflow.keras.layers import Convolution1D, MaxPooling1D
from keras.layers.recurrent import LSTM, GRU
from keras import regularizers
import theano
theano.config.compute_test_value = "ignore"



def shuffle_in_unison(a, b):
    # courtsey http://stackoverflow.com/users/190280/josh-bleecher-snyder
    assert len(a) == len(b)
    shuffled_a = np.empty(a.shape, dtype=a.dtype)
    shuffled_b = np.empty(b.shape, dtype=b.dtype)
    permutation = np.random.permutation(len(a))
    for old_index, new_index in enumerate(permutation):
        shuffled_a[new_index] = a[old_index]
        shuffled_b[new_index] = b[old_index]
    return shuffled_a, shuffled_b
```

```
def create_Xt_Yt(X, y, percentage=0.9):
    p = int(len(X) * percentage)
    X_train = X[0:p]
    Y_train = y[0:p]

    X_train, Y_train = shuffle_in_unison(X_train, Y_train)

    X_test = X[p:]
    Y_test = y[p:]

    return X_train, X_test, Y_train, Y_test

def shuffle_in_unison(a, b):
    assert len(a) == len(b)
    shuffled_a = np.empty(a.shape, dtype=a.dtype)
    shuffled_b = np.empty(b.shape, dtype=b.dtype)
    permutation = np.random.permutation(len(a))
    for old_index, new_index in enumerate(permutation):
        shuffled_a[new_index] = a[old_index]
        shuffled_b[new_index] = b[old_index]
    return shuffled_a, shuffled_b

def create_Xt_Yt(X, y, percentage=0.8):
    p = int(len(X) * percentage)
    X_train = X[0:p]
    Y_train = y[0:p]

    X_train, Y_train = shuffle_in_unison(X_train, Y_train)

    X_test = X[p:]
    Y_test = y[p:]

    return X_train, X_test, Y_train, Y_test

data = pd.read_csv("data.csv")
data = data.loc[:, '2668.000000.1'].pct_change().dropna().tolist()[-10000:]
```

```
plt.style.use('seaborn-darkgrid')
plt.title('S&P500')
plt.plot(data, color = 'green')
plt.show()

window = 30
emb_size = 1
step = 1
forecast = 5
X, Y = [], []
for i in range(0, len(data), step):
    try:
        x_i = data[i:i+window]
        y_i = data[i+window+forecast]

        last_close = x_i[window-1]
        next_close = y_i

        if last_close < next_close:
            y_i = [1, 0]
        else:
            y_i = [0, 1]

    except Exception as e:
        print(e)
        break

    X.append(x_i)
    Y.append(y_i)

X = [(np.array(x) - np.mean(x)) / np.std(x) for x in X]
X, Y = np.array(X), np.array(Y)

X_train, X_test, Y_train, Y_test = create_Xt_Yt(X, Y)
```

```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], emb_size))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], emb_size))



model = Sequential()
model.add(Convolution1D(input_shape = (window, emb_size), kernel_initializer = 'he_normal',
                        filters = 32,
                        kernel_size = 2,
                        padding = 'same',
                        #subsample_length = 2,
                        kernel_regularizer = regularizers.l2(0.01),
                        activity_regularizer = regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(ELU())
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(64))
model.add(BatchNormalization())
model.add(ELU())
model.add(Dense(2))
model.add(Activation('softmax'))
opt = Nadam(lr=0.002)
reduce_lr = ReduceLROnPlateau(monitor='val_acc', factor=0.9, patience=25, min_lr=0.000001, verbose=1)
checkpointer = ModelCheckpoint(filepath = "cnn.hdf5", verbose = 1, save_best_only = True)

model.compile(optimizer=opt,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

print(model.summary())

history = model.fit(X_train, Y_train,
          epochs = 100,
          batch_size = 128,
          verbose=1,
          validation_data=(X_test, Y_test),
```

```
        callbacks=[reduce_lr, checkpointer],
        shuffle=True)

plt.style.use('seaborn-darkgrid')
plt.figure(figsize = (10.5, 7))
plt.plot(history.history['loss'], color = 'seagreen')
plt.plot(history.history['val_loss'], color = 'tomato')
plt.title('model loss', fontsize=24, fontweight='bold')
plt.ylabel('loss', fontsize=17, fontweight='bold')
plt.xlabel('epoch', fontsize=17, fontweight='bold')
plt.legend(['train', 'test'], loc='best', fontsize=17)
plt.show()

plt.figure(figsize = (10.5, 7))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy', fontsize=24, fontweight='bold')
plt.ylabel('accuracy', fontsize=17, fontweight='bold')
plt.xlabel('epoch', fontsize=17, fontweight='bold')
plt.legend(['train', 'test'], loc='best', fontsize=17)
plt.show()

model.load_weights("cnn.hdf5")
pred = model.predict(np.array(X_test))

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

C = confusion_matrix([np.argmax(y) for y in Y_test], [np.argmax(y) for y in pred])
print(classification_report([np.argmax(y) for y in Y_test], [np.argmax(y) for y in pred]))
C = C / C.astype(np.float).sum(axis=1)
print('True positive = ',  round(C[0][0], 3))
print('False positive = ',  round(C[0][1], 3))
print('True negative = ',  round(C[1][1], 3))
print('False negative = ',  round(C[1][0], 3))
```

```
#LSTM

model = Sequential()
model.add(LSTM(window, input_dim = 1,
               return_sequences = False,
               recurrent_dropout = 0.9,
               kernel_regularizer = regularizers.l2(0.01),
               activity_regularizer = regularizers.l2(0.01)))
model.add(Dropout(0.7))
model.add(Dense(64))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(2))
model.add(Activation('softmax'))
opt = Nadam(lr=0.002)
reduce_lr = ReduceLROnPlateau(monitor='val_accuracy', factor=0.9, patience=25, min_lr=0.000001, verbose=1)
checkpointer = ModelCheckpoint(filepath = "lstm.hdf5", verbose = 1, save_best_only = True)
model.compile(optimizer=opt,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

print(model.summary())

history = model.fit(np.expand_dims(X_train, axis=-1), Y_train,
          epochs = 50,
          batch_size = 128,
          verbose=1,
          validation_data=(X_test, Y_test),
          callbacks=[reduce_lr, checkpointer],
          shuffle=True)

plt.style.use('seaborn-darkgrid')
plt.figure(figsize = (10.5, 7))
plt.plot(history.history['loss'], color = 'seagreen')
plt.plot(history.history['val_loss'], color = 'tomato')
plt.title('model loss', fontsize=24, fontweight='bold')
```

```
plt.ylabel('loss', fontsize=17, fontweight='bold')
plt.xlabel('epoch', fontsize=17, fontweight='bold')
plt.legend(['train', 'test'], loc='best', fontsize=17)
plt.show()

plt.figure(figsize = (10.5, 7))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy', fontsize=24, fontweight='bold')
plt.ylabel('accuracy', fontsize=17, fontweight='bold')
plt.xlabel('epoch', fontsize=17, fontweight='bold')
plt.legend(['train', 'test'], loc='best', fontsize=17)
plt.show()

model.load_weights("lstm.hdf5")
pred = model.predict(np.array(X_test))

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

C = confusion_matrix([np.argmax(y) for y in Y_test], [np.argmax(y) for y in pred])
print(classification_report([np.argmax(y) for y in Y_test], [np.argmax(y) for y in pred]))
C = C / C.astype(np.float).sum(axis=1)
print('True positive = ',  round(C[0][0], 3))
print('False positive = ',  round(C[0][1], 3))
print('True negative = ',  round(C[1][1], 3))
print('False negative = ',  round(C[1][0], 3))
```